The BASIC Programmer's Toolkit
for the Commodore PET personal computer
1979-1983

by Len Shustek, 7 November 2023

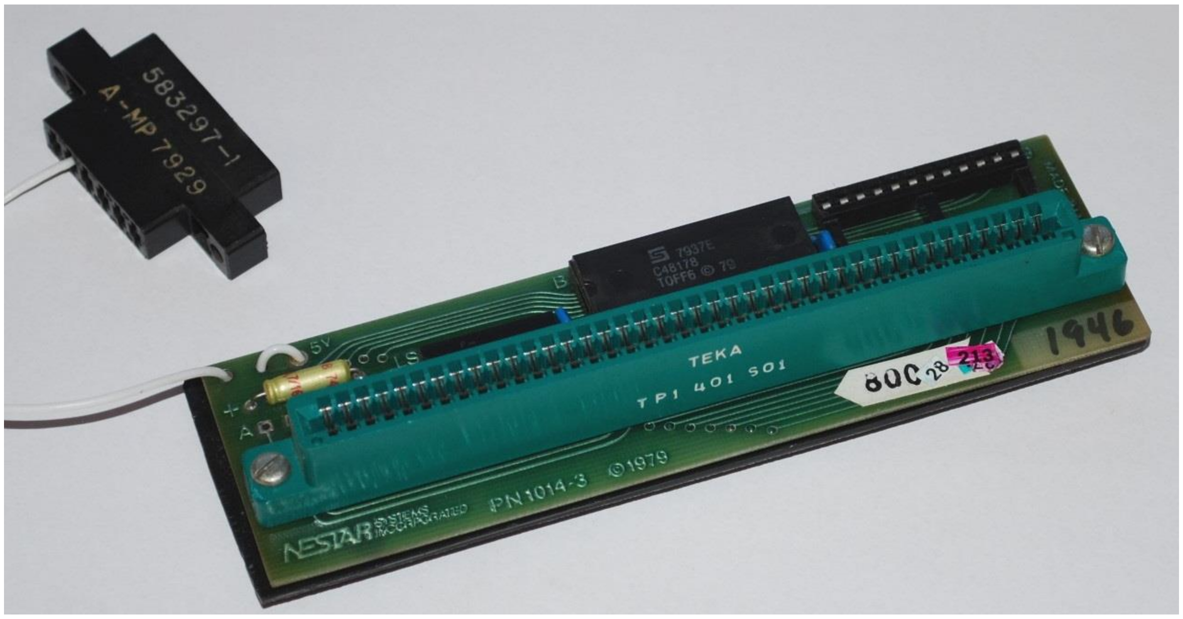
The 1977 Commodore PET, https://en.wikipedia.org/wiki/Commodore_PET, was one of the three most popular early consumer personal computers, along with the Radio Shack TRS-80 and the Apple][.

The PET came with a BASIC language interpreter that Commodore licensed from Microsoft. Although the language was adequate, the program editing and debugging tools were not.

In 1978 programmer Chuck Bond devised a set of programming aids for PET BASIC. He approached Harry Saal and Len Shustek of the fledging networking company Nestar Systems, https://en.wikipedia.org/wiki/Nestar_Systems, to see if they were interested in making it a product. They were, but Shustek insisted (Appendix 7) that it be done under a different name so as not to dilute the networking brand they were trying to establish. In August 1979 Nestar created a wholly-owned subsidiary called "Palo Alto ICs" that began selling the "BASIC Programmer's Toolkit".

The Toolkit was an unauthorized extension to the BASIC interpreter that added much needed features -- such as program renumbering and search -- to the built-in BASIC in ROM. The integration was entirely reverse-engineered, with no support from either Commodore or Microsoft. Nonetheless, since it enhanced the PET's BASIC and made their computer more attractive, Commodore was supportive.

The Toolkit was supplied as an add-on ROM. For the original version of the PET it was on a small custom PC board that plugged into the memory expansion port.



For the updated PET, it was supplied as an IC that plugged into a spare socket on the motherboard. The suggested retail prices were $79.95 and $49.95, respectively.

In both cases the Toolkit seamlessly added the following ten commands to the BASIC interpreter that came with the PET:

command	function
AUTO	Provides new line numbers when you are entering BASIC program lines.
RENUMBER	Renumbers your BASIC program, including all GOTOs and GOSUBs.
DELETE	Removes groups of BASIC program lines.
FIND	Locates and displays the BASIC program lines that contain a specified string.
APPEND	Adds a previously SAVEd program to the one currently in your PET.
DUMP	Displays the names and values of all the variables used by your program (excluding arrays).
HELP	If your program stops due to an error, HELP displays the offending line and shows where the PET detected the error.
TRACE	As a program runs, the last six line numbers being executed are shown in the upper right corner of the PET's screen.
STEP	Executes one BASIC line and stops. Pressing SHIFT executes the next line. The line number is displayed in the upper right corner of the screen.
OFF	Turns TRACE or STEP off.

The manual (Appendix 1) was written by Gregory Yob, a computer game designer whose 1975 *Hunt the Wumpus* was one of the earliest adventure games.

The Toolkit was packaged (Appendix 2) in boxes and shrink-wrap bags for retail distribution and was sold internationally through dealers, supported by marketing materials and advertisements (Appendix 3) from Palo Alto ICs .

It was favorably reviewed (Appendix 4) in trade magazines and newsletters. Some of the praise was effusive; one said, “it may change your life”.

The Toolkit became extremely popular. In the first six months there were over a million dollars of retail sales, which inspired the production of unauthorized copies by counterfeiters. In a February 1980 draft press release (Appendix 5) that was reminiscent of Bill Gates’ famous 1976 open letter to hobbyists, Palo Alto ICs noted the million-dollar milestone but decried the illegal copies, saying “the substantial investment necessary to produce quality software will not be made if illegal pirating becomes widespread.”

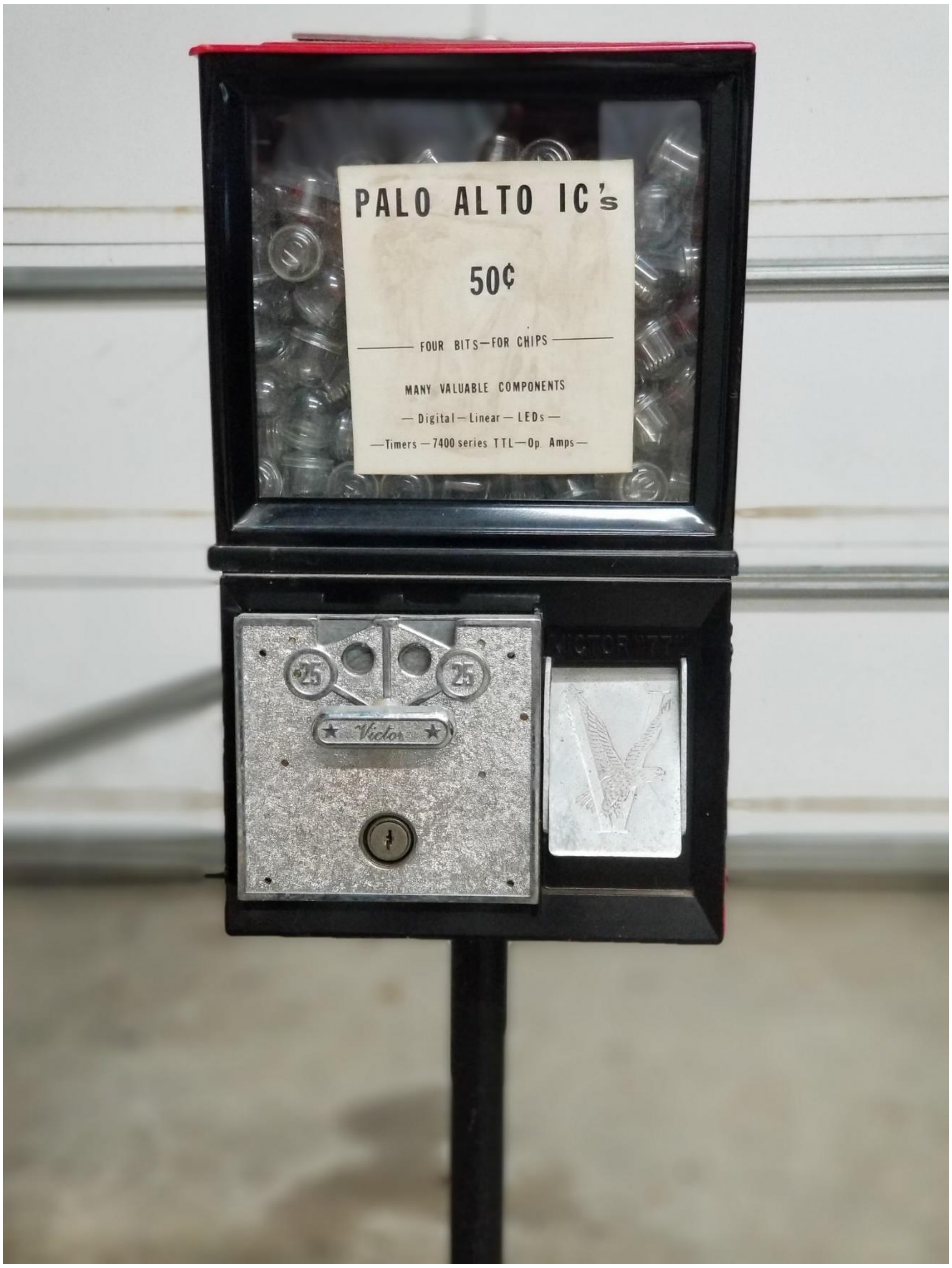
After two years, over 25,000 units had been shipped. (Appendix 6).

The original 6502 assembly-language source code of the Toolkit has been lost. But in In 2008 the original author, Chuck Bond, reconstructed the source code from a disassembly of the ROM; see it at NestarSystems/BASIC_Programmer's_Toolkit at main · LenShustek/NestarSystems · GitHub.

The manual for the BASIC Programmer’s Toolkit is in the collection of the Smithsonian’s National Museum of American History (https://americanhistory.si.edu/collections/search/object/nmah_1372236) and is installed in the PET that is also in their collection.

An amusing coda

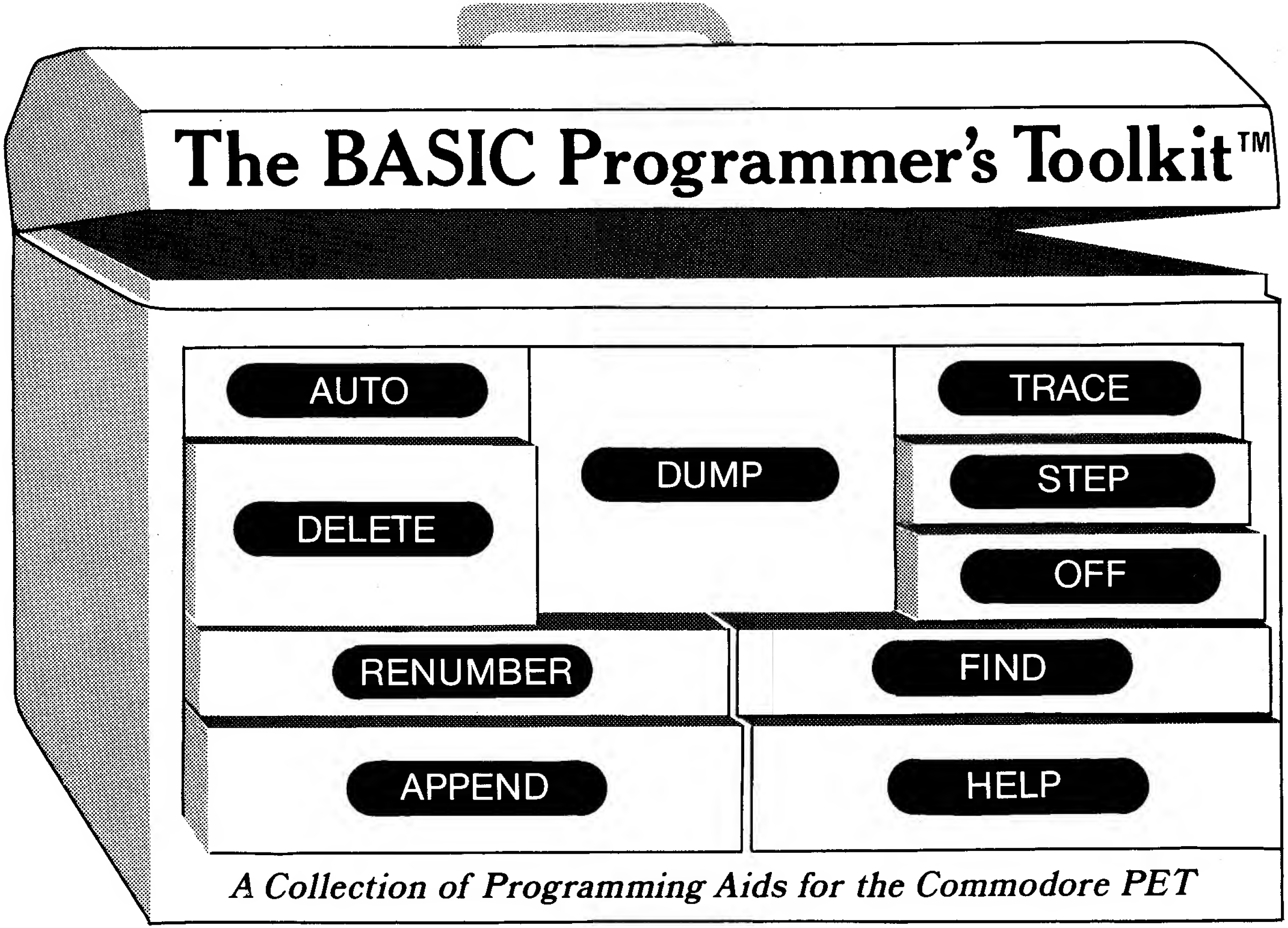
The only other offering of Palo Alto IC's was a vending machine that was installed in the Palo Alto Byte Shop computer store. It dispensed a random integrated circuit chip in a plastic container for 50 cents. It is unclear whether any of the ICs were ever purchased.



Appendix 1

BASIC Programmer’s Toolkit manual

The BASIC Programmer's Toolkit™



A Collection of Programming Aids for the Commodore PET

THE BASIC PROGRAMMER'S TOOLKIT

A
Collection of Programming Aids
for the Commodore PET

USER'S GUIDE

Chuck Bond is the creator and main implementor of the BASIC Programmer's Toolkit.

This user's guide was originally written by Gregory Yob.

Revision 1   August 1979

What is it?

Your new BASIC Programmer's Toolkit^{tm} is a machine language program which is provided in a 2 Kilobyte ROM. When this is installed in your PET, and the Toolkit's program activated, your PET's BASIC has ten new and very useful commands

AUTO	Provides new line numbers when you are entering BASIC program lines.
RENUMBER	Renumbers your BASIC program, including all GOTOs and GOSUBs.
DELETE	Removes groups of BASIC program lines.
FIND	Locates and displays the BASIC program lines that contain a specified string.
APPEND	Adds a previously SAVEd program to the one currently in your PET.
DUMP	Displays the names and values of all the variables used by your program (excluding arrays).
HELP	If your program stops due to an error, HELP displays the offending line and shows where the PET detected the error.
TRACE	As a program runs, the last six line numbers being executed are shown in the upper right corner of the PET's screen.
STEP	Executes one BASIC line and stops. Pressing SHIFT executes the next line. The line number is displayed in the upper right corner of the screen.
OFF	Turns TRACE or STEP off.

Getting Started

INSTALLATION

There are several versions of the Commodore PET -- models 2001-4, 2001-8, 2001-16, and 2001-32, and the PET's main circuit board has changed in many ways. For our purposes, though, there are only two kinds of PET, the "old" and the "new". The "old" PET has a small keyboard and the tape unit on the front, and on the right side there is a Memory Expansion Port. This port consists of PC fingers to which a connector can be attached. The "new" PET has a large keyboard, a separate tape unit which connects to the tape socket in the rear of the PET, and the Memory Expansion Port is made up of many pins that extend upwards from the main circuit board.

Please follow the instructions appropriate to your PET. If you have some expansion memory, such as the Expandamem, contact your dealer for installation information.

TOOLKIT INSTALLATION FOR "OLD" PETS

Your Programmer's Toolkit is mounted on a 1½" by 5" PC card which has an edge connector for attaching to the Memory Expansion Port. A short wire with a small connector extends from the Toolkit PC card and attaches to the Cassette Port in the back of the PET.

As shown in the diagram, plug the Toolkit onto the Memory Expansion Port with the ICs on the PC card facing the PET, and the wire extending to the back. Then plug the wire's connector into the 2nd Cassette Port with the wire towards the corner of the PET. The connector on the end of the wire is polarized so that it cannot be plugged in upside down.

BE SURE THE PET'S POWER IS OFF WHEN YOU ATTACH OR REMOVE YOUR TOOLKIT!!!

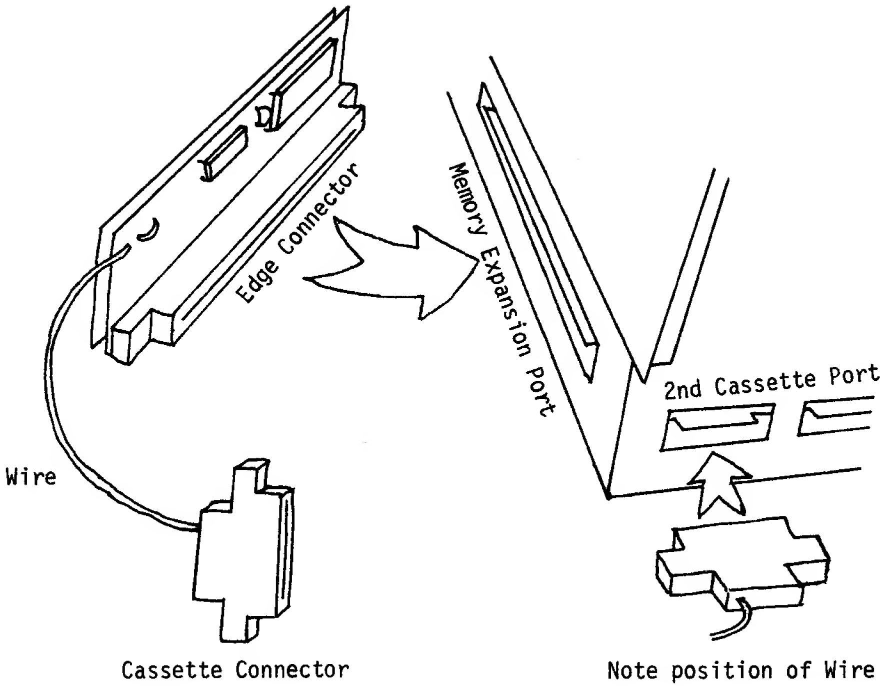


Figure 1. Installation for "OLD" PETS

TOOLKIT INSTALLATION FOR "NEW" PETS

For the new PET your Programmer's Toolkit consists only of the ROM IC with the Toolkit's program inside. If you have by some mischance got a Toolkit for the "Old" PET, check with your dealer. In most cases the chip from the "Old" PET version of the Toolkit cannot be used in the "New" PETs.

If you haven't opened your PET and looked inside, or if you have never installed or removed a 24 pin IC, YOU ARE STRONGLY ADVISED TO HAVE SOMEONE WHO HAS DONE THESE THINGS INSTALL THE TOOLKIT FOR YOU!

Turn the power off and remove the power cord from the wall socket. PLEASE DO THIS!

Unscrew the four screws on the bottom of the top half of the PET (about 5" back from the front of the PET on each side). Use the correct size phillips-head screwdriver to avoid damaging the screws.

SLOWLY open the PET's case by lifting the front. Inside will be several cables, some of which might be too short to allow full opening of the PET. Find these cables and gently disconnect them from the PET's main circuit board. When the case is fully open, set the bar (on the left side) so that the case will remain opened.

In the PET there is a row of 7 ROM sockets, with three vacancies on the right side. As shown in the diagram, install the Toolkit IC next to the PET's ROM ICs, with the notch and dot as shown. BE SURE THE NOTCH AND DOT ARE PLACED CORRECTLY!!

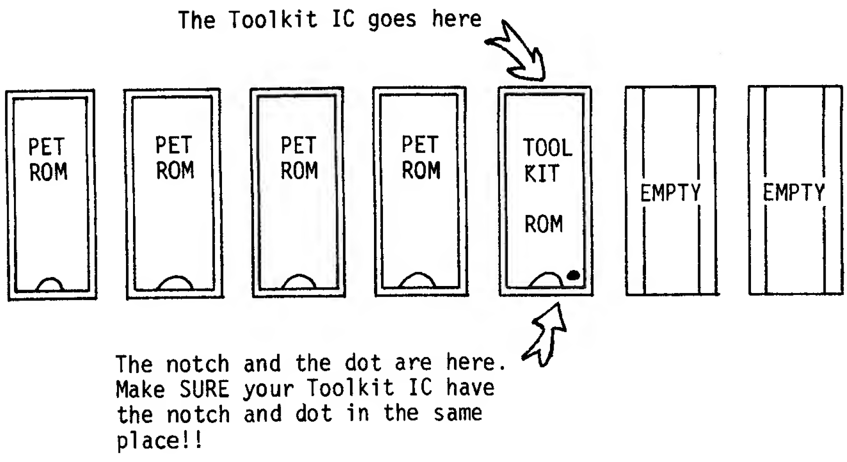

The Toolkit IC goes here
PET ROM
PET ROM
PET ROM
PET ROM
TOOL KIT ROM
EMPTY
EMPTY
The notch and the dot are here. Make SURE your Toolkit IC have the notch and dot in the same place!!

Figure 2. Installation for "NEW" PETS

SOME CAUTIONS ON INSTALLING ICs: 1) Touch the metal case of the PET with your hand before installing the Toolkit IC. This is to discharge any static electricity that might damage the ICs. 2) Remove the IC from the black plastic carrier that it has been shipped in. 3) CAREFULLY place the IC in the socket and make sure all the pins are aligned with the socket's receptacles. This may take a little cautious fiddling. 4) Press SLOWLY and FIRMLY on the IC until it slides completely into the socket. 5) Check for bent pins -- especially for a pin that has bent UNDER the Toolkit IC.

If you have bent any pins, remove the IC carefully using a small blunt knife or a metal nail file. Using needlenosed pliers, straighten the pins, and try installation again. You can only do this once or twice before the pins break, so get it right the first time!!!

After the IC is installed and checked, reconnect any cables that were disconnected, close the PET, and replace the four screws.

S T A R T U P & C H E C K O U T

Make sure you have installed your Toolkit correctly, turn on your PET, and then enter either

```
SYS 45056
```

or

```
SYS 11*4096
```

and press RETURN.

Your PET should reply with:

```
(C) 1979 PAICS

READY.
```

If this does not happen, TURN THE POWER OFF IMMEDIATELY and check your installation. If you have an "Old" PET, check that the 2nd Cassette connector is correctly placed and that the Toolkit board is completely seated onto the printed circuit board. "New" PET owners should check the position of their IC and the direction of the notch and dot.

If the Toolkit message still does not appear after you have checked your installation, contact your dealer for assistance.

After the Toolkit has been initialized with the SYS command, the new commands are available for use. The remainder of this manual describes the commands in detail.

The Commands

AUTO (First Line Number) , (Line Number Interval)

AUTO provides automatic line numbers when you are entering BASIC program lines. The first line number will be displayed, and when the BASIC line is entered via RETURN, the line number interval is added and the new line number will be displayed.

To leave AUTO mode, just press RETURN. If you don't provide a first line number or line number interval in the command, AUTO assumes that you meant AUTO 100,10. AUTO will remember the last line number and interval if a previous AUTO was executed.

EXAMPLES:

Type in NEW to remove any programs in your PET, and then type:

```
AUTO
```

The PET will respond with:

```
100 ▧            (The ▧ indicates the PET's cursor.)
```

The line number 100 will appear, and the cursor is in the right position for entry of a BASIC line. You may enter as many lines as you like, each terminated with RETURN. To leave AUTO mode, press RETURN without entering a line.

```
AUTO
 100 REM ONCE UPON A TIME
 110 REM A FAIRY PRINCESS
 120 REM MET MY PET COMPUTER.
 130
▧
```

You are now out of AUTO mode, and can do other things, like RUN. AUTO line numbering can be resumed where you left off just by typing AUTO again.

```
RUN

READY.
AUTO
 130 █
```

AUTO always remembers the last line number it gave to you.

If you want to start with a new line number, you can provide it on the AUTO command:

```
AUTO 456
 456 REM THEN SHE SMILED AT IT
 466 █
```

To change the line number interval requires that you provide both the first line number and the interval:

```
AUTO ,15

?SYNTAX ERROR                (It didn't work!)
READY.

AUTO 1000,15
1000 █
```

If you change only the first line number, AUTO remembers the old interval:

```
AUTO 2000
 2000 REM AND ASKED 'WHO ARE YOU?'
 2015
```

There are some situations in which AUTO will behave in an unexpected manner. The section on GOTCHAS describes these unusual cases.

RENUMBER (First Line Number) , (Line Number Interval)

RENUMBER renumbers the entire program currently in your PET. RENUMBER will change all line numbers, including those in IF - THEN, GOTO, GOSUB, ON-GOTO, ON-GOSUB, RUN, and LIST. References to non-existent line numbers are changed to 63999.

If no First Line Number or Line Number Interval are given, RENUMBER assumes you meant RENUMBER 100,10.

EXAMPLES:

Enter a small program on your PET:

```
NEW
10 REM A RENUMBERISH EXAMPLE
20 GOTO 10
30 GOSUB 1000
40 IF K=2 THEN 500
1000 REM SAMPLE SUBROUTINE
1002 RETURN

RENUMBER

READY.
LIST
 100 REM A RENUMBERISH EXAMPLE
 110 GOTO 100
 120 GOSUB 140
 130 IF K=2 THEN 63999
 140 REM SAMPLE SUBROUTINE
 150 RETURN
```

The GOTO in Line 20 and the GOSUB in Line 30 have been changed to reflect their new line numbers. Line 40 now jumps to 63999 since the program did not have a Line 500 in it

(Note: After you RENUMBER a program, you can then find all the illegal line number references by using FIND 63999.)

To start the line numbers with 5000 instead of 100, just use:

```
RENUMBER 5000
```

```
READY.
LIST

 5000 REM A RENUMBERISH EXAMPLE
 5010 GOTO 5000
  (etc...)
```

If a different line number interval is desired, you must provide the starting line number:

```
RENUMBER ,3

?SYNTAX ERROR

RENUMBER 300,3

READY.
LIST

 300 REM A RENUMBERISH EXAMPLE
 303 GOTO 300
  (etc...)
```

If the last new line number in your program is going to be larger than 63999, RENUMBER will tell you so:

```
RENUMBER 50000,10000
?OUT OF RANGE ERROR
```

No part of your program will have been renumbered if you get an ?OUT OF RANGE ERROR.

DELETE (Line Number) - (Line Number)

DELETE removes BASIC lines by specifying the line number range in the same way that LIST specified the BASIC lines to be displayed. For example, DELETE 100-200 will remove all lines from 100 to 200. DELETE -100 will remove all lines from 0 to 100. DELETE 100- removes lines 100 to the end of the program.

EXAMPLES:

```
NEW

10 REM ONE LINE FOR THIS TIME
20 REM ANOTHER TO FILL THE BILL
30 REM AGAIN TO SATISFY THE YEN
40 REM OK, KEEP IT THIS WAY
50 REM FLY ALONG, GET A LITTLE HIGH
60 REM DON'T STUMBLE WHEN YOU MUMBLE
DELETE 30-45

READY.
LIST
 10 REM ONE LINE FOR THIS TIME
 20 REM ANOTHER TO FILL THE BILL
 50 REM FLY ALONG, GET A LITTLE HIGH
 60 REM DON'T STUMBLE WHEN YOU MUMBLE
```

DELETE's line number range works like LIST -- there doesn't have to be a line at the numbers you specify.

DELETE won't work without some line numbers. This prevents the loss of the entire program by mistake. Use NEW to delete the entire program

```
DELETE

?SYNTAX ERROR

DELETE 50

READY.
LIST

 10 REM ONE LINE FOR THIS TIME
 20 REM ANOTHER TO FILL THE BILL
 60 REM DON'T STUMBLE WHEN YOU MUMBLE.
```

APPEND (Program Name)

APPEND will load a previously saved program from cassette tape and add it to the end of the program already in your PET's memory. The APPEND command does not interleave or overwrite the program in the PET.

The program name works in the same way that LOAD does. You can provide either "name" or a string containing "name" and the PET will search the tape until it finds a program with the same initial characters as "name". You may also specify cassette unit 1 or 2, exactly as with the LOAD command.

EXAMPLES:

Enter this small program and SAVE it:

```
200 REM THIS IS PROGRAM ONE          (Be sure to NEW
210 REM TO SHOW HOW APPEND            to remove any
220 REM WORKS ON YOUR PET             other program.)
SAVE "FIRST PROGRAM"
```

Now remove this program and enter another one:

```
NEW

READY.
100 REM THIS IS ANOTHER PROGRAM
110 REM TO CONTINUE THE EXAMPLE.
```

Rewind your FIRST PROGRAM tape, and now add it via APPEND:

```
APPEND

PRESS PLAY ON TAPE #1
OK

SEARCHING
FOUND FIRST PROGRAM
APPENDING
```

The PET looks for a program just like LOAD does, and when the program is found, APPENDING appears to let you know what's going on.

```
LIST

 100 REM THIS IS ANOTHER PROGRAM
 120 REM TO CONTINUE THE EXAMPLE.
 200 REM THIS IS PROGRAM ONE
 210 REM TO SHOW HOW APPEND
 220 REM WORKS ON YOUR PET.
```

You can repeat APPEND as often as you like until memory is full. Another APPEND of FIRST PROGRAM results in:

```
LIST

 100 REM THIS IS ANOTHER PROGRAM
 110 REM TO CONTINUE THE EXAMPLE.
 200 REM THIS IS PROGRAM ONE
 210 REM TO SHOW HOW APPEND
 220 REM WORKS ON YOUR PET.
 200 REM THIS IS PROGRAM ONE
 210 REM TO SHOW HOW APPEND
 220 REM WORKS ON YOUR PET.
```

APPEND simply adds the program on tape to the <u>end</u> of the current program. It <u>does</u> <u>not</u> insert or overwrite lines within the current program.

If you have several routines stored on a tape, you can select the ones you want by mentioning their names -- just like LOAD does. Suppose your tape contains:

```
APPLE
PEAR
APPLICATION
PEACH
```

APPEND "APPLE" will find the first program -- so will the use of APPEND "APP". APPEND "PEAC" finds the program PEACH.

Since APPEND disregards line numbers, it is your responsibility to make sure your APPENDed programs have increasing line numbers. When BASIC executes IF - THEN or GOTO / GOSUB, the line numbers are searched from the start of the program until a line number equal to or larger than the jump's number is found. This means that the program won't be able to find out-of-order line numbers and you will see an ?UNDEF'D STATEMENT ERROR.

FIND (BASIC code) , (Line Number) - (Line Number)

FIND "(string)" , (Line Number) - (Line Number)

FIND locates and displays all lines that contain a specified fragment of BASIC code or a quoted string constant. The line numbers select the lines to be searched in the same way that LIST selects lines to be displayed on the screen. If the line numbers are omitted, the entire program is searched.

If the item being searched for is not a string surrounded by quotation marks, then the BASIC program itself is searched, excluding any quoted strings. If the item being searched is surrounded by quotes, then only quoted strings within the program will be searched. For example, FIND A will find all occurrences of the variable A such as A=3.1415, and FIND "A" will find the character A inside strings such as PRINT "MOVE AGAIN".

Note that BASIC programs are stored internally as "chunks" of text called tokens, and not just as sequences of characters. Each BASIC keyword, such as PRINT and THEN, is a separate token, as are special characters such as = and +. When FIND looks at BASIC text, tokens must match entirely in order for the search to be successful. Thus in order to find, say, all PRINT statements in a program, you must type FIND PRINT, and not just FIND PRI. This often works to your advantage, however, because you can also type FIND I to find all occurrences of the variable I without also finding all PRINT statements. Of course, you can also look for sequences of tokens, as in FIND I = or FIND IF X=0.

When you use FIND in a large program, more than a screen full of lines might be listed. Just as for LIST, you may press RVS to slow the display, and STOP to stop it entirely.

EXAMPLES:

Here is a short program to use as an example:

```
10 PRINT" [clear screen][4 cursor down][4 cursor right]
     THIS IS A SILLY"
20 PRINT" [4 cursor right] EXAMPLE FOR YOUR"
30 PRINT" [4 cursor right] SERIOUS CONSIDERATION."
40 FOR J=1 TO 1000: NEXT J
50 INPUT" [5 cursor down] TRY IT AGAIN";A$
60 IF A$="YES" THEN 10
70 IF A$="NO" THEN END
80 PRINT" [clear screen] ":GOTO 40
```

Let's try a few BASIC code searches:

```
FIND IF
 60 IF A$="YES" THEN 10
 70 IF A$="NO" THEN END

READY.
```

Notice that FIND prints the entire line if the search is successful. You can then use the screen editor to change the lines as needed and reenter them.

The line numbers can be used to search portions of your program. These work in exactly the same way that LIST does.

```
FIND A$,70-
 70 IF A$="NO" THEN END
```

Remember to use quotes to search inside strings. For example:

```
FIND FOR
 40 FOR J=1 TO 1000: NEXT J

FIND "FOR"
 20 PRINT" [4 cursor down] EXAMPLE FOR YOUR"
```

The FOR without quotes looked for tokens, and the search ignored the FOR inside the quotes in line 20. The "FOR" with quotes searcned in the quoted parts of the program only.

You can search for cursor movements and graphics characters by putting them inside quotation marks:

```
FIND "[cursor down][cursor right]"
 10 PRINT" [clear screen][4 cursor down][4 cursor right]
     THIS IS A SILLY"
```

Sometimes it is important to distinguish lines from other similar lines. To do so, just FIND with a more complex program segment; it need not be a single item.

```
FIND THEN 10
 60 IF A$="YES" THEN 10
```

There are two lines with the keyword THEN, so using the item THEN 10 made the search more precise. When making complex items, be careful with blanks.

```
FIND THEN10

READY
```

Your item must match the blanks in the program lines as well as the non-blank parts.

DUMP

DUMP displays all of the non-array variables present in the PET's memory. The variables are displayed in the form:

(variable) = (value)

which permits the use of the screen editor to change their values.

When a program has many variables in it, the DUMP display can be "frozen" with the SHIFT key. STOP will exit the DUMP entirely.

The variables are displayed in the order that they were created.

EXAMPLES:

Execute the following direct statements on your PET:

```
CLR
X=3                              (This example ignores the
Y=2                               PET's reply of READY)
A$="HELLO OUT THERE"
C%=256
```

To see these variables, use DUMP:

```
DUMP
X=3
Y=2
A$="HELLO OUT THERE
C%=256
```

Notice that the variables are displayed in the order that these were created. DUMP will always display the current value for each variable:

```
Y=123456987
Z=555666777
DUMP
X=3
Y=123456987
A$="HELLO OUT THERE"
C%=256
Z=555666777
```

Try using the screen editor to change these values, like making A$ become "SOMETHING ELSE NOW". When you DUMP again, the values will be changed.

When a lot of variables are in the PET, the screen will become filled and the first variables will scroll off the top of the screen. For example:

```
CLR
A=1:B=2:C=3:D=4:E=5:F=6:G=7:H=8:I=9:J=10
K=11:L=12:M=13:N=14:O=15:P=16:Q=17:R=18
S=19:T=20:U=21:V=22:W=23:X=24:Y=25:Z=26
DUMP

......
W=23
X=24
Y=25
Z=26

READY.
```

Variables A - D scrolled off the top of the screen. Try DUMP again, and press the SHIFT key a moment later. The display will stop, and remain "frozen" as long as SHIFT remains depressed. By releasing SHIFT for short moments you can look at a DUMP in groups of 2 or 3 variables.

If you press STOP, the DUMP will be aborted:

```
DUMP
A=1
B=2                        (Press STOP immediately after DUMP!)
C=3
D=4

BREAK
READY
```

You can combine SHIFT and STOP to let the display move to the variables that interest you. Use SHIFT to find the variable, and then press STOP to terminate the DUMP.

The Programmer's Toolkit does not DUMP array variables for two reasons. First, DUMPing arrays is much more difficult to do in machine language, especially with 2 and 3 dimensional arrays. Second, in most cases only a few array elements are of interest when debugging a program, and the rest just clutter up the screen. If you want to display an array, use FOR-NEXT in a direct command:

```
FOR J=Ø TO 2Ø:PRINT A(J):NEXT
```

DUMP provides a very convenient way to display the value of string variables that contain cursor-motion characters. Since the string values are displayed in quotes, the cursor-motion characters are printed as their reverse-graphic equivalents just as they are entered in a program, and they may be easily changed using the screen editor. In contrast, printing the value of a string containing cursor-motion characters causes the cursor motion to occur, which makes it difficult to see what the string actually contains.

HELP

When a program is RUN and an error is encountered, the PET will print an error message and the line number. The HELP command will print the line on the screen and indicate where the PET found an error by printing the "bad" spot in reversed field.

HELP must be used <u>immediately</u> after an error, or else the PET "forgets" the location of the error. (HELP will do nothing if any other command is used first.)

If a program is interrupted via the STOP key, HELP will display the line that includes the last successfully completed statement. The reversed field indicator will be at the end of the last completed statement.

EXAMPLES:

Here is a program with some bugs in it:

```
10 X=10/0
20 Y=((((((15)))))))
30 Z=123456789123456789123456789123456789123456789

RUN

?DIVISION BY ZERO ERROR IN 10   (These examples ignore
HELP                             the PET's READY. message.)
 10 X=10/0
```

The zero is in reverse field (indicated here with an underline) which is where the PET decided something was wrong.

```
GOTO 20

?SYNTAX ERROR IN 20
HELP
 20 Y=((((((15)))))))
```

If you carefully count the parentheses, the reversed field indicator is on the 6th right parenthesis. Since there were 6 left parentheses going in, the 7th right parenthesis should have made the error. In most cases, HELP puts the indicator at the character <u>before</u> the suspected error.

Try GOTO 30 and see where the PET gives up.

Lines displayed by HELP are easily changed and reentered by using the screen editor as usual.

The PET easily forgets where errors were because that information is kept in temporary storage. If _any command at all_ is done before HELP, the HELP line is forgotten.

```
RUN

?DIVISION BY ZERO ERROR IN  10
PRINT X
 0

HELP

READY
```

The PRINT X made the PET forget the location of the error, and HELP isn't much help now.

T R A C E

The TRACE command turns on a "tracer" which will display the currently executed line number when a program is RUN. The last six line numbers are displayed in the upper right corner of the screen in a reverse-field "window". As lines are executed, their numbers scroll up the "window" with the most recently executed line's number at the bottom.

Pressing SHIFT when a program is running will slow the TRACE down to about 2 lines per second. Even without the SHIFT key, though, TRACE slows the running of a program considerably.

EXAMPLES:

Enter this program into your PET:

```
NEW
10 PRINT"[clear screen]";
20 X=1
30 PRINT"[home cursor]"X
40 X=X+1
50 GOTO 30
```

Now trace it:

```
TRACE

READY.
RUN
```

The screen will clear, and a reverse-field "window" appears in the upper right corner. In the upper left, a number is shown which counts upward. A typical display might look like this:

```
13                                  #50
                                    #30
                                    #40
                                    #50
                                    #30
                                    #40
```

(On the PET, the numbers starting with # will be in reverse-field -- black on white.)

Press the SHIFT key and notice how the display slows down. Now about 2 lines per second appear in the "window".

To stop the program, press the STOP key. STOP will work with SHIFT depressed also.

Try pressing RETURN when you have stopped this program with TRACE. You will get a ?SYNTAX ERROR. Whenever the RETURN key is pressed, the PET scans the entire line -- and when you stopped this program, the cursor was in the 5th line on the screen. On the right, the "window" is still there, and the PET took the #(line number) as your entry.

If you leave TRACE with the cursor in the top 6 lines of the screen, press [cursor down] or [clear screen] to remove the "window" first.

Let's try another program:

```
NEW
10 PRINT"[home cursor]"X:X=X+1:GOTO 10
RUN
```

When this is TRACEd, only line 10 appears in the window -- and only on the bottom line. The TRACE does not fill the window with the line number if the program is in a loop on one line. This was done intentionally since many programs have "GET" loops like this one:

```
222 GET A$: IF A$="" THEN 222
```

TRACE will let you see what happened before the loop by not repeating the 222 in the window.

STEP

The STEP command activates the "tracer" and then executes one BASIC line and stops. (The line number is displayed in the "window" in the upper right corner of the PET's screen.) To execute the next line, simply press SHIFT.

If you hold SHIFT down, the program will continue to execute lines until SHIFT is not depressed. To exit a program, press the STOP key; this will work regardless of whether the SHIFT key is pressed.

After RUN, SHIFT must be pressed to execute the first line. If a loop is included in a program line, like:

```
10 GOTO 10
```

the line will be executed indefinitely. Press STOP to leave the program.

EXAMPLES:

Here is a small program to demonstrate STEP:

```
NEW
10 PRINT"FIRST THING"
20 PRINT"SECOND ITEM"
30 PRINT"THIRD OBJECT"
40 GOTO 10
STEP

READY.
RUN
```

At this point, the "window" will appear in the upper right corner of the PET's screen with line #10 at the bottom. Note that line #10 has <u>not</u> been executed yet.

Tap the SHIFT key briefly. FIRST THING will appear, and the "window" now contains #10 and #20. Each time you press SHIFT, STEP will execute one BASIC line and display the line number of the next statement that will be executed.

Hold the SHIFT key down. Now the "window" will scroll the line numbers in the same way that TRACE does, and the screen will show:

```
FIRST THING                         #20
SECOND ITEM                         #30
THIRD OBJECT                        #40
FIRST THING                         #10
                                    #20
(etc)
```

If you press STOP, the program will end, and this works with or without SHIFT depressed. If the cursor is at the bottom of the screen when STOP is pressed, only the last three lines of the "window" will be visible because the PET has to scroll the screen to display the BREAK IN ### and READY. messages.

STEP works like TRACE if a one-line loop is encountered:

```
NEW
10 PRINT"[home cursor]"X:X=X+1:GOTO10
RUN
```

STEP is waiting for you to press SHIFT. Press SHIFT and release it.

The "window" remains the same, with only the #10 shown. However, an increasing number appears near the upper left corner as the increasing X is displayed. When STEP sees a one-line loop, the loop is executed until STOP is pressed. The "window" behaves like TRACE, and does not fill up with repetitions of the same line number.

O F F

OFF turns off the "tracer" which TRACE and STEP activate. After OFF, a running program will no longer display the "window" or wait for the SHIFT key before executing the next line.

EXAMPLES:

Take the example program for STEP and RUN it with STEP turned on. Press the STOP key, and then type OFF.

Now RUN the program again -- the "window" is gone, and the program RUNs as usual.

Gotchas!

Your Programmer's Toolkit has been designed to give you the most powerful and effective extensions to your PET's BASIC for a reasonable price. The Toolkit takes advantage of the PET's ROM subroutines wherever possible, and as a result, the PET's "way of doing things" is reflected in the Toolkit.

By now, you know that a computer does what it is told to do, and that is not always what you meant for it to do. There are many situations where the Toolkit commands will act in a "strange" way -- that is, the Toolkit's action will not be what you expected.

These situations have been called "Gotchas!", and this part describes the Gotchas! that we know about. Take the time to try these situations out, and then the Toolkit won't have nasty surprises for you in the middle of an important program.

INSTALLATION

The old model PETs have the Toolkit installed on the Memory Expansion Port on the right side of the PET. A short wire and connector are provided which supplies +5 volt power from the 2nd Cassette Port. Be sure both connectors are installed properly.

The new PET's Toolkits are installed on one of the PET's ROM sockets on the Main Logic Board. Be sure that you are using the right socket and that Pin 1 of your Toolkit is in Pin 1 of the socket.

PETs with expansion memories that use the Memory Expansion Port will require differing arrangements. Expandamem users need a small ROM board (available from your dealer) which fits on the Expandamem's expansion sockets.

Contact your dealer if you have any questions concerning the installation of your Toolkit.

MEMORY

The Programmer's Toolkit requires ROM space for the machine language program, and a small amount of RAM to remember variable date. The memory space used is:

```
$ Ø3EØ - $ Ø3FF  for RAM
$ BØØØ - $ B7FF  for ROM
```

The RAM is in the upper part of the 2nd Cassette Buffer, and with the exception of APPEND, use of the 2nd Cassette in BASIC programs with the Toolkit may have unpredictable results.

The ROM is placed above the Screen RAM and will not interfere with most PET users, including those with memory expansion RAM. If you have a Computhink Disk, a conflict exists. The PC board which comes with the "old PET" version of the Toolkit has an extra socket which can be used for any B2716-compatible ROM or PROM; it is wired for addresses starting at $9000.

Initialization

The SYS 45056 (or SYS11*4096) command is necessary to initialize the Toolkit and activate its commands. You need to do it again only if the PET has been turned off, or reset with a switch, or if a machine language routine has been used to reset the PET software.

SOME GENERAL FEATURES

Immediate Mode Only

Your Toolkit's commands will work as direct statements only. If you include a Toolkit command in a BASIC program, you will get a ?SYNTAX ERROR.

Toolkit Commands Only

You must make a Toolkit command the only item in a direct statement. You will get a ?SYNTAX ERROR if you try to put more than one command on a line, or if you try to mix Toolkit and BASIC statements on the same line.

The Toolkit will either ignore extra characters, or, in most cases, give you a ?SYNTAX ERROR. If you do one thing at a time with your Toolkit, there won't be any trouble.

PET Abbreviations Work

As most of you know, the PET will accept shortened versions of the BASIC keywords if the last character is shifted. For example, L[shift-O] will perform a LOAD, V[shift-E] will do a VERIFY, and so on.

Your Toolkit will accept abbreviated commands as well. Here is a list of Toolkit abbreviations. The underlined letter is shifted:

AU	AUTO
RE	RENUMBER
DE	DELETE
FI	FIND
AP	APPEND
DU	DUMP
HE	HELP
TR	TRACE
ST	STEP
OF	OFF

Other partial abbreviations will work -- for example, RE, REN, and RENUM, will all function correctly.

AUTO

When using AUTO, you often want to use the screen editor on previously entered lines and then return to AUTO. AUTO will remember the next line number if the line number of the edited line is less than the next line number that AUTO is to provide.

If a larger line number is edited, the AUTO's line number is changed to reflect this. Here is an example to get you started; there are many combinations -- try several to see how this works.

```
AUTO 100,10
 100 REM LINE ONE
 110 REM LINE TWO
 120
```

Now use the screen editor to change Line 100 to 200 and press RETURN. AUTO now provides the new line number, 210, where 110 used to be.

When leaving AUTO after entering lines in the middle of a program, be aware that pressing RETURN after a line number results in the deletion of a line! Use the DEL key to remove the line number, and <u>then</u> press RETURN on the blank line if you are in danger of losing a line.

AUTO 100,0 and AUTO 0,0 are acceptable to the Toolkit, though rather useless.

If the next line that AUTO is to provide is larger than 63999, you will get an ?OUT OF RANGE ERROR, and the last line you entered will be missing. For example,

```
AUTO 50000,10000
 50000 REM ONE
 60000 REM TWO
?OUT OF RANGE ERROR
READY.
LIST

 50000 REM ONE
READY.
```

RENUMBER

RENUMBER always renumbers the entire program in memory. If a line number is unreferenced, e.g., there is a GOTO xxx and the program does not have line xxx, the line number is changed to 63999. Use FIND 63999 to locate these references to missing lines.

If the largest line number after RENUMBER is to be more than 63999, RENUMBER gives an ?OUT OF RANGE ERROR and won't renumber the program.

RENUMBER 200,0 and RENUMBER 0,0 will work -- and leave you with a mess!

When RENUMBER converts small line numbers to large ones, and vice versa (e.g., 25 to 1000), the program has to be moved in the PET memory and the line linkage pointers changed. A large program that is drastically renumbered will take some time to do. Programs that are extremely long (i.e., under 100 bytes free) may get an ?OUT OF MEMORY ERROR and not be renumbered correctly. One cure is to start with a fresh copy of the program and RENUMBER 1,1 to keep the line numbers short. But you probably won't have enough space for variables when you try to run the program anyway!

Numbers included in REM and in quoted strings won't be changed by RENUMBER. Be aware of this if you have hidden a statement like

```
REM GOTO 500 - CONDITIONAL FOR 16K PET
```

or similar nasty deeds.

Programs that modify themselves by printing statements on the screen and then stuffing the input buffer with RETURNs will probably not work correctly after RENUMBERing.

If you have APPENDed a program with out-of-order line numbers, RENUMBER can be used to make the code accessible by the program. Any jumps (GOTO, IF-THEN, etc.) will now be pointed to incorrect line numbers, so beware!

DELETE

DELETE without any line numbers, or DELETE -, will give a ?SYNTAX ERROR. This is intended to prevent loss of your program through an accidental DELETE. (To remove a program, use the NEW command.)

If DELETE is given out-of-order line numbers, it will give a ?SYNTAX ERROR if a line exists in the given (but backwards) range. If there are no lines, DELETE does nothing.

FIND

The line number range for FIND operates like LIST. FIND without a line number range looks at the entire program. If line numbers are given out-of-order, FIND behaves like DELETE.

If the search item is not in quote marks, it is tokenized, and the search proceeds through the unquoted parts of the BASIC program only. If the search item is surrounded by quotes, only the quoted parts of the BASIC program are searched.

Remember that FIND without a quoted string looks for matching text that has been broken into tokens. Since all of the BASIC program text has been tokenized, FIND PRINT will look for and find all PRINT statements. However, REM statements are not tokenized; PRINT inside a REM is stored as a sequence of five characters instead of a single token. The result is that FIND PRINT will not find a statement like REM PRINT RESULTS, because it is looking only for the token PRINT. One way out of this difficulty is to search only for partial keywords in REM statements; as in: FIND RINT.

FIND FOO "BAZ" will tokenize FOO and ignore "BAZ". A similar fate awaits FIND "FOO" BAZ, with the search being for "FOO". FIND searches only the part of the line after the line number, and will not find the line number itself.

APPEND

APPEND works for cassettes #1 and #2 only. APPEND will not recognize IEEE device numbers, and will not work for disk.

If the program to be appended would exceed the available memory, APPEND will abort. The program size is on the tape header, so partially appended programs will not result.

Tapes are searched for program names in the same way that LOAD does.

DUMP

DUMP will show the variables whenever BASIC still has them. Note that editing any line in the program causes all the variables to be discarded.

HELP

HELP is very transient, and must be the first command after stopping a program, or nothing will be displayed.

IF YOU DISCOVER A NEW GOTCHA!

PAICs would like to know about any bugs, etc. concerning the Programmer's Toolkit. Please write us with the required details to duplicate the bug. However, please don't phone. Though phone conversations are nice for the psyche, they take time away from making new and neater products, and, phone calls tend to be forgotten when it is time to fix or improve a product. So, please write instead.

The reverse field marker is often one character before the offending part.

When a program is stopped by STOP, or ends by itself, HELP will display the last executed line. The reverse field marker will be at the end of the last completed statement.

The reverse field marker will reverse the entire token in HELP.

If the first character in a line is in error, HELP will not display a reverse field marker.

In some cases, the source of the error might not be where the marker is; the marker is what the PET was looking at when the error was discovered. This is especially true in arithmetic expressions and with READ or INPUT or GET.

TRACE

If the cursor is in the upper six lines of the screen after TRACE has been used, any direct command will include the characters in the "window" when you press RETURN -- which won't work very well. Either clear the screen, or move the cursor below the window on the screen, and then enter your commands. Shift-RETURN will move the cursor down nicely.

STEP

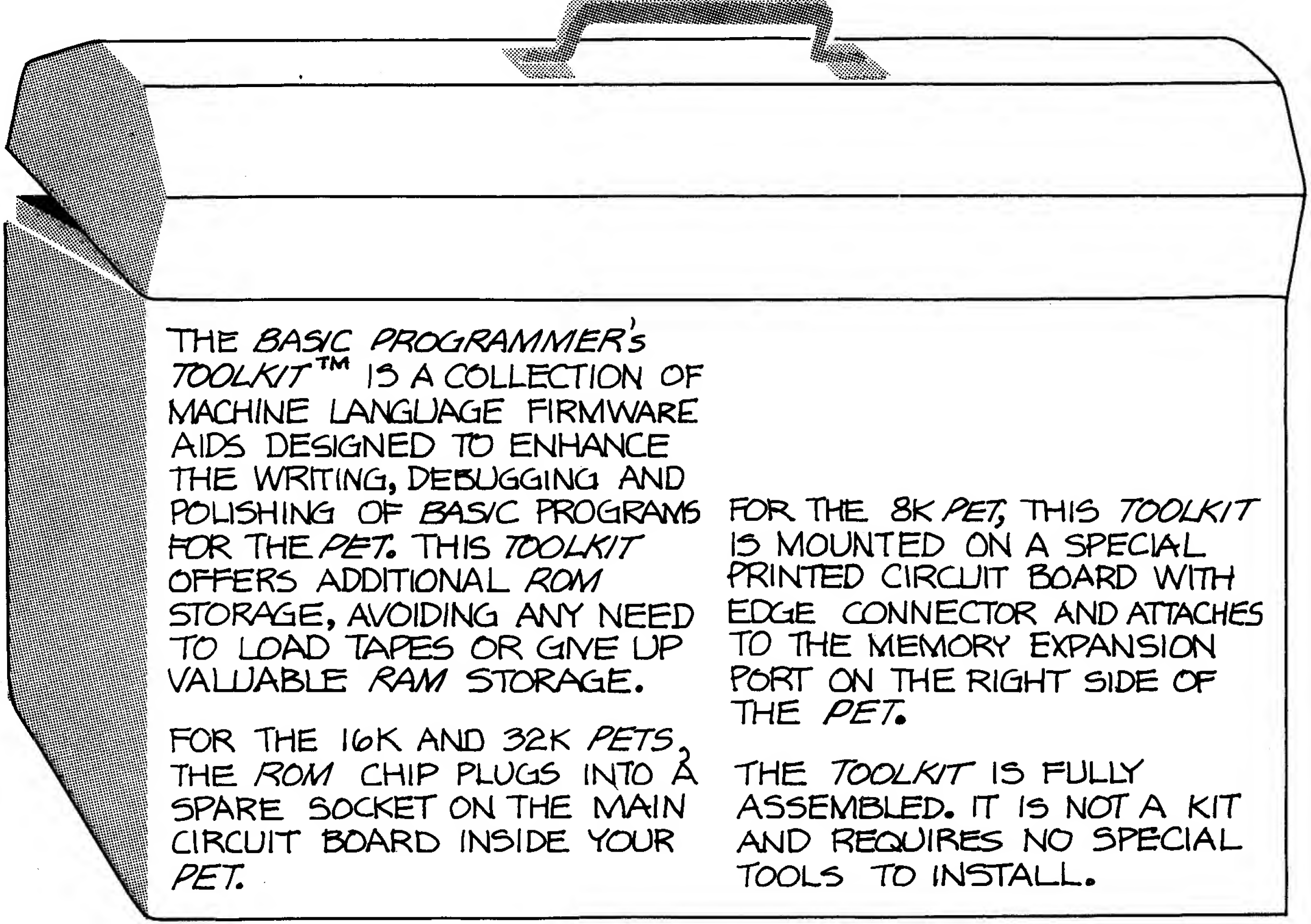
Same as TRACE.

OFF

No known GOTCHAS!

CAUTION FOR HACKERS

The Toolkit requires that the PET BASIC program be intact and correct regarding pointers, line numbers, etc. If you have mangled a program, the Toolkit might go bonkers.



THE *BASIC PROGRAMMER'S TOOLKIT*™ IS A COLLECTION OF MACHINE LANGUAGE FIRMWARE AIDS DESIGNED TO ENHANCE THE WRITING, DEBUGGING AND POLISHING OF *BASIC* PROGRAMS FOR THE *PET*. THIS *TOOLKIT* OFFERS ADDITIONAL *ROM* STORAGE, AVOIDING ANY NEED TO LOAD TAPES OR GIVE UP VALUABLE *RAM* STORAGE.

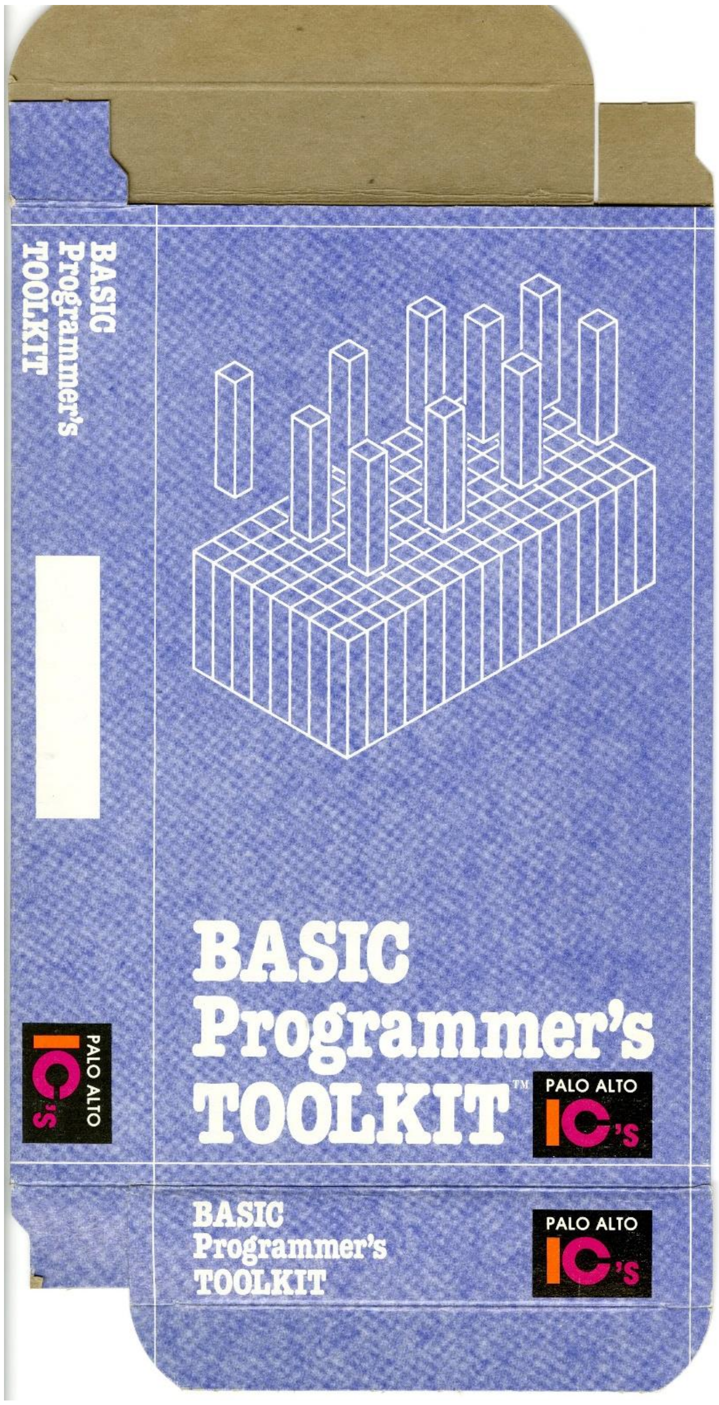
FOR THE 16K AND 32K *PETS*, THE *ROM* CHIP PLUGS INTO A SPARE SOCKET ON THE MAIN CIRCUIT BOARD INSIDE YOUR *PET*.

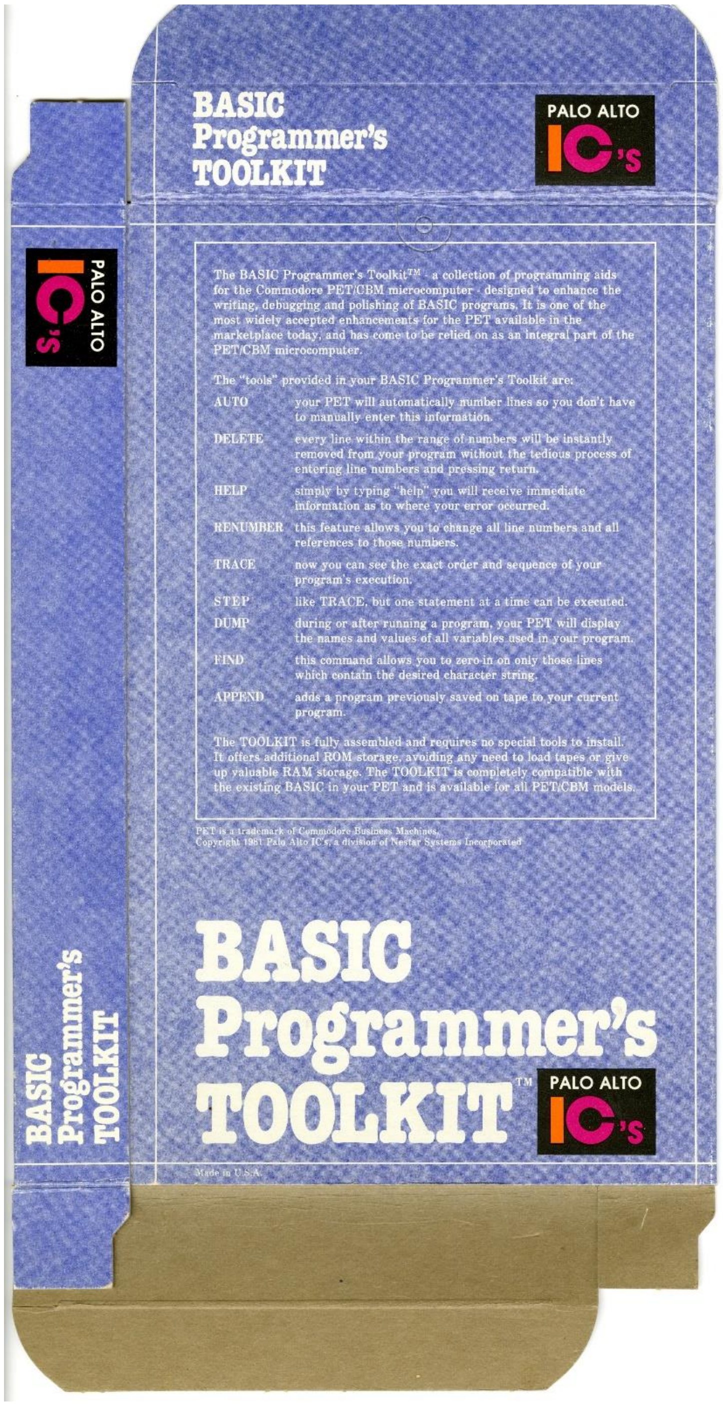
FOR THE 8K *PET*, THIS *TOOLKIT* IS MOUNTED ON A SPECIAL PRINTED CIRCUIT BOARD WITH EDGE CONNECTOR AND ATTACHES TO THE MEMORY EXPANSION PORT ON THE RIGHT SIDE OF THE *PET*.

THE *TOOLKIT* IS FULLY ASSEMBLED. IT IS NOT A KIT AND REQUIRES NO SPECIAL TOOLS TO INSTALL.

Appendix 2

BASIC Programmer's Toolkit packaging


BASIC Programmer's TOOLKIT
PALO ALTO IC's
BASIC Programmer's TOOLKIT™
PALO ALTO IC's
BASIC Programmer's TOOLKIT
PALO ALTO IC's


BASIC Programmer's TOOLKIT
PALO ALTO IC's
PALO ALTO IC's
BASIC Programmer's TOOLKIT
BASIC Programmer's TOOLKIT™
PALO ALTO IC's
Made in U.S.A.

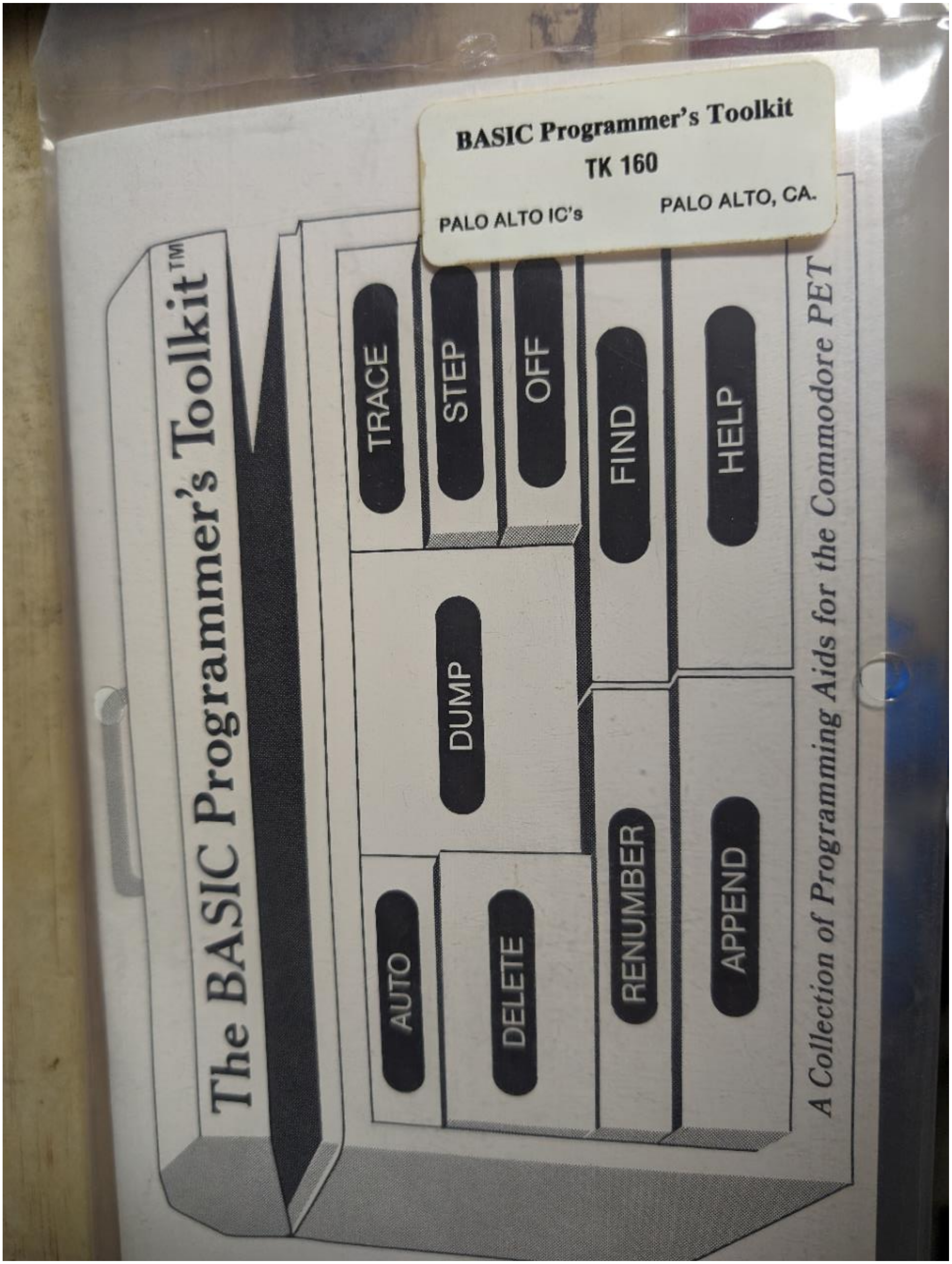
The BASIC Programmer's Toolkit™ - a collection of programming aids for the Commodore PET/CBM microcomputer - designed to enhance the writing, debugging and polishing of BASIC programs. It is one of the most widely accepted enhancements for the PET available in the marketplace today, and has come to be relied on as an integral part of the PET/CBM microcomputer.

The "tools" provided in your BASIC Programmer's Toolkit are:

AUTO	your PET will automatically number lines so you don't have to manually enter this information.
DELETE	every line within the range of numbers will be instantly removed from your program without the tedious process of entering line numbers and pressing return.
HELP	simply by typing "help" you will receive immediate information as to where your error occurred.
RENUMBER	this feature allows you to change all line numbers and all references to those numbers.
TRACE	now you can see the exact order and sequence of your program's execution.
STEP	like TRACE, but one statement at a time can be executed.
DUMP	during or after running a program, your PET will display the names and values of all variables used in your program.
FIND	this command allows you to zero-in on only those lines which contain the desired character string.
APPEND	adds a program previously saved on tape to your current program.

The TOOLKIT is fully assembled and requires no special tools to install. It offers additional ROM storage, avoiding any need to load tapes or give up valuable RAM storage. The TOOLKIT is completely compatible with the existing BASIC in your PET and is available for all PET/CBM models.

PET is a trademark of Commodore Business Machines.
Copyright 1981 Palo Alto IC's, a division of Nestar Systems Incorporated


BASIC Programmer's Toolkit

TK 160

PALO ALTO IC's PALO ALTO, CA.

The BASIC Programmer's Toolkit™

A Collection of Programming Aids for the Commodore PET

Appendix 3

BASIC Programmer’s Toolkit marketing materials

PALO ALTO ICs

810 Garland Drive
Palo Alto, California 94303

WARNING
TO ALL PET OWNERS
THE NEW BASIC PROGRAMMER'S TOOLKIT™ WILL NOT NECESSARILY MAKE YOU A BETTER PERSON. NO CLAIM—IMPLIED OR OTHERWISE—IS MADE. BUT THE BASIC PROGRAMMER'S TOOLKIT MAY VERY WELL MAKE YOU A BETTER PROGRAMMER. AT THE VERY LEAST IT WILL MAKE PROGRAMMING YOUR PET MORE FUN. *THIS* IS BOTH AN IMPLICIT AND EXPLICIT CLAIM.

An open letter to PET owners:

I love my PET as much as you love yours. But don't you sometimes find it frustrating to create or modify programs?

I've owned my PET for over a year; I'm impressed with its features and its power. Yet when I program I find a great deal of difficulty in modifying, polishing, simplifying and adding new features.

But now, with the BASIC PROGRAMMER'S TOOLKIT™ that I and my associates have created, I'm having more fun, making far fewer errors and I'm completing programs much faster.

I'd like to share my discovery with you.

The BASIC PROGRAMMER'S TOOLKIT has two kilobytes of ROM firmware on a single chip and contains a collection of machine language programs available from the time you turn on your PET to the time you shut it off. Your PET will now have a significant set of additional commands—programmer's tools to enhance your ability to create new programs, to modify old programs, to discover how programs work . . . to enjoy your PET even more. And to do it all so much faster and so much more accurately.

There are no tapes to load or to interfere with any running programs. And it installs in minutes, without tools.

Once you have your BASIC PROGRAMMER'S TOOKIT in place, you'll know how much more pleasure your programming can give you. I know: I've been using the TOOLKIT for almost two months. I think I love it as much as I love my Pet.

Cordially,
PALO ALTO ICs

[illegible]

Harry J. Saal
President

A Subsidiary of NESTAR SYSTEMS, INC.

The Tools in the Basic Programmer's Tool Kit™

VERY SPECIAL COMMANDS. Power up your PET, execute the command **SYS** with the appropriate address and your *new* PET is off and running with these commands:

AUTO This command is followed by a series of optional parameters specifying where you want to enter lines and how far apart you want them. Your PET will automatically respond with a line number prompt. You won't have to enter the line numbers; you won't have to worry about errors of screen editing.

```
AUTO 100,25
   100 FOR I=1 TO 10
   125 GOSUB 300
   150 ■
```

DELETE Like **LIST**, this command is followed by a range of line numbers . . . and every one of the lines within the range of numbers will be removed instantly from your program. No longer will you have to type a line number, press **RETURN**, type the next line number, **RETURN**, next line. . . .

```
DELETE 200-350

READY.
LIST 200-350

READY.
```

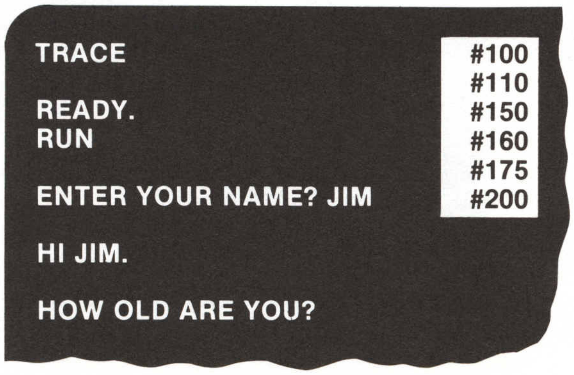
RENUMBER Now you can change all line numbers— and all references to those numbers—instantly, as, for instance by evenly spaced increments of 100 or 25 or 10. . . .

```
LIST

   10 GOSUB 99
   15 PRINT I
   16 GOTO 10
   99 INPUT J
  100 IF J=0 THEN END
  200 I=SQR(J):RETURN
READY

RENUMBER 100,10

READY.
LIST

  100 GOSUB 130
  110 PRINT I
  120 GOTO 100
  130 INPUT J
  140 IF J=0 THEN END
  150 I=SQR(J):RETURN
READY.
```

HELP How many times have you wanted to scream "HELP!" when your PET couldn't interpret your program and all it would say was: **?SYNTAX ERROR**? Fred no longer: Now just type in **HELP**. The line on which the error occurs will be shown and the erroneous portion of the line will be indicated in reverse video on the screen. Truly a great help in any learning or school situation.

```
RUN

 ?DIVISION BY ZERO ERROR IN 500
 READY.
 HELP
  500 J=SQR(A*B/C)

 READY
```

TRACE Now you can see precisely the order and sequence in which your program is being executed. You can also stop the program at any point and record the sequence. Type in this command and your PET will keep a record of the line numbers of the last six statements executed. These last six statement numbers will appear in a small rectangular window in the upper right hand corner of your screen.

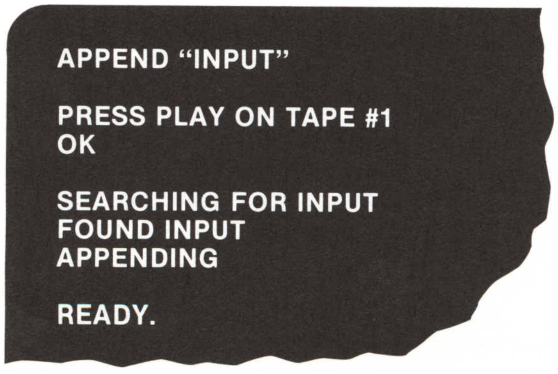


```
TRACE                                  #100
                                       #110
READY.                                 #150
RUN                                    #160
                                       #175
ENTER YOUR NAME? JIM                   #200

HI JIM.

HOW OLD ARE YOU?
```

STEP Again your line numbers are displayed in the upper right hand corner of the screen in this version of **TRACE**. But now your PET executes just one statement and pauses until you press **SHIFT**. Then it proceeds to the next statement.

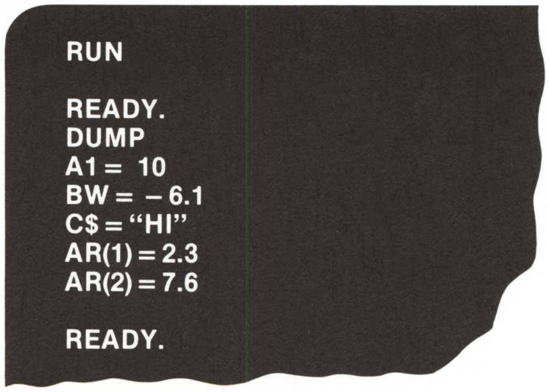
OFF This command will stop either **TRACE** or **STEP**.

APPEND You've already worked hard to develop a number of programs and, of course, you've saved them on tape. You're working on a program that's now in memory. Just type in **APPEND "***program name***"** and all statements in that program will now follow the program in memory. No need to retype; no opportunity for errors. Save any set of statements or subroutines onto a tape, using the normal **SAVE** command; then recall them with the **APPEND** command and add them permanently to your program.



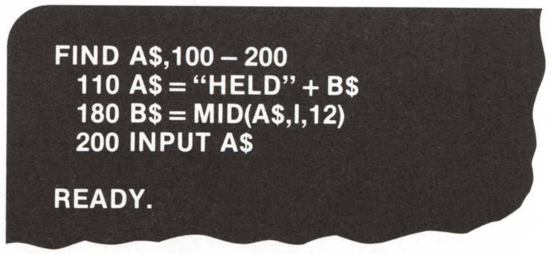
```
APPEND "INPUT"

PRESS PLAY ON TAPE #1
OK

SEARCHING FOR INPUT
FOUND INPUT
APPENDING

READY.
```

DUMP During or after running a program, this command will display the names and values of all variables used in the execution of your program. You'll know at once what simple variables, arrays and strings you've used and what values you've assigned to each.



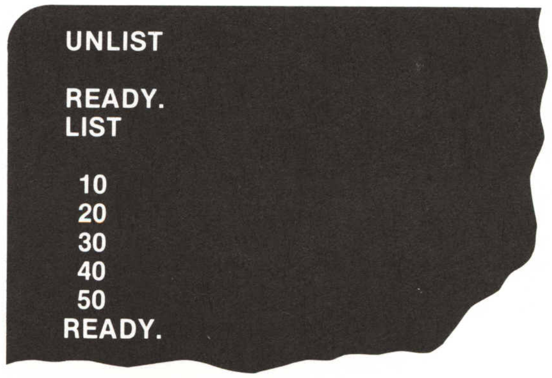
```
RUN

READY.
DUMP
A1 =  10
BW = −6.1
C$ = "HI"
AR(1) = 2.3
AR(2) = 7.6

READY.
```

FIND Like **LIST**, this command will show a set of lines. But the FIND command is followed by specifying a character string. Those lines, and only those lines, containing a desired character string will be listed on you PET's screen. If you were to type **FIND A$, 100-500** your PET's screen would display all lines between line numbers 100 and 500 that contain **A$**.



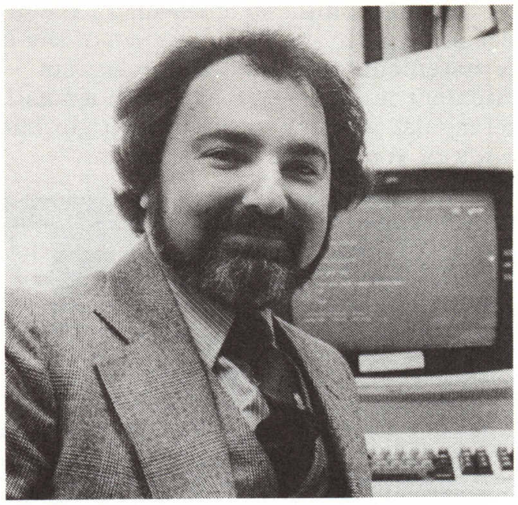
```
FIND A$,100 – 200
  110 A$ = "HELD" + B$
  180 B$ = MID(A$,I,12)
  200 INPUT A$

READY.
```

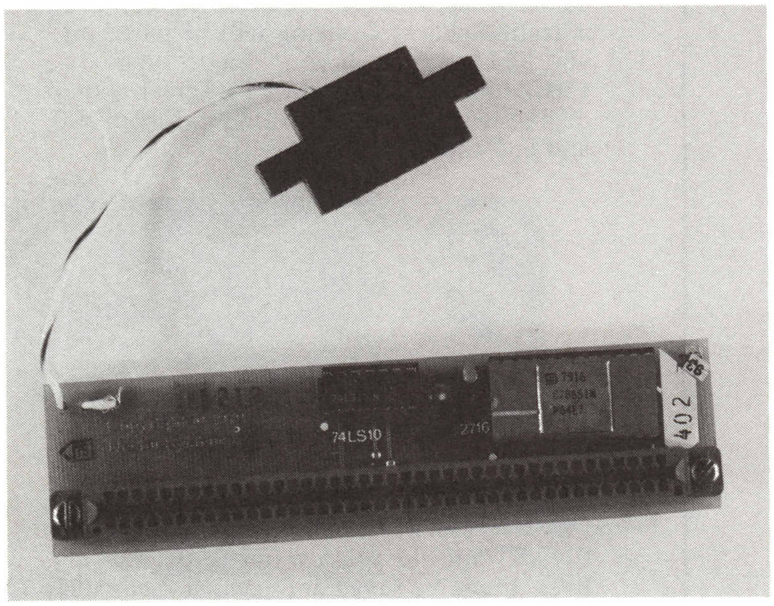
UNLIST An extraordinary command; very necessary in certain applications. Type **UNLIST** and the text cannot be displayed. Ever. It can be saved and used subsequently on any PET but neither you nor anyone else can ever type **LIST** and see the program. It provides confidentiality when necessary . . . as in answers to a test. . . .



```
UNLIST

READY.
LIST

 10
 20
 30
 40
 50
READY.
```

HARRY SAAL, with 20 years of computing experience, has become a leading expert in the field of personal computers. He holds a PhD in Physics from Columbia University, has taught Computer Science at two major universities and has been a researcher and systems programmer with IBM. He is currently president of Nestar Systems, Inc., and its subsidiary, Palo Alto ICs, both devoted to the development of personal computer peripherals and systems. The Nestar System is considered by experts to be the ultimate multiple microcomputer program storage system.





THE BASIC PROGRAMMER'S TOOLKIT™

Two KB of ROM firmware on a single chip. For PET 2001-8, the 8K RAM PET, the printed circuit board shown is attached, with the connectors shown, to the memory expansion interface at the right side of your PET and to the second cassette interface located at the rear right of your PET.

Memory addresses in ROM from \$98ØØ to \$9FFF and addresses in RAM from \$Ø3EØ to \$Ø3FF are used by the TOOLKIT.

For the 16K and 32K PET, the BASIC PROGRAMMER'S TOOLKIT consists of the ROM firmware chip which is to be installed in the right-most empty socket inside the PET.

The TOOLKIT is custom designed to interface, at your request, with either the Skyles or ExpandaPet memory expansion systems. The TOOLKIT may also be ordered for interfacing with the 8K PET but without connectors. Please indicate when ordering.

A comprehensive user's manual is supplied at no additional charge.

PALO ALTO ICs A Division of Nestar Systems, Incorporated

430 SHERMAN AVENUE
PALO ALTO, CALIFORNIA 94306
415/327-0125

DEALER INFO

The Toolkit is available in several configurations to accommodate the different needs of PET owners:

Part Number		Description
TK-80		Single chip ROM (2316 type) for the 8K PET—(suitable for use with BETSI Memory Expansion).
TK-80P		Same chip on PC Board with connectors (uses second cassette connector and memory expansion connector).
TK-80E	same pricing as TK-80P	Same chip on PC Board for Expandamem.
TK-80S		Same chip on PC Board for Skyles Memory Expansion.
TK-160		Single chip ROM (2316 type) for 16K or 32K PET with new ROM chip set.

Note: When you turn on the PET, if you see ***** COMMODORE BASIC***** the PET has "old" ROMs and needs a TK-80 (P, E, or S) Toolkit.

If you see **### COMMODORE BASIC ###** the PET has "new" ROMs and needs a TK-160 Toolkit.

ADDRESSING:
The Toolkit occupies memory locations $B000 through $B7FF. Any memory expansion or peripheral attached to the PET, may result in possible incompatibility with the Toolkit. (Not compatible with Computhink disk system.)

GENERAL:
The Toolkit provides a set of additional commands for the PET. Installation instructions are included with each Toolkit.

The APPEND command applies to the first and second cassette units only—*not* the IEEE port or disk peripherals.

The presence of the Toolkit will not interfere with other operations of the PET.

EXCHANGE:
PET Owners who have purchased a TK-80 (P, E, or S) Toolkit may exchange the ROM for a TK-160. The charge for exchange is $15.00 to the customer; $10.00 to dealer.

Thank you for your interest in the BASIC Programmer's Toolkit.

PALO ALTO ICs A Division of Nestar Systems, Incorporated

430 SHERMAN AVENUE
PALO ALTO, CALIFORNIA 94306
415/327-0125

CONFIDENTIAL DEALER PRICES

(Effective September 1, 1979)

BASIC Programmer's Toolkit™

Model	**Description**	**Suggested Retail**	**Quantity 10-24**	**Quantity 25 up**
TK-80	ROM only—for 8K PET ROMs.	$49.95	$35.00	$33.50
TK-80P (E or S)	ROM for 8K PET, plus PCB connectors, etc., to install on memory expansion connector.	$79.95	$56.00	$53.50
TK-160	ROM to install inside 16K or 32K PETs which have new PET ROMs.	$49.95	$35.00	$33.50

Each Toolkit™ complete with explanatory manual on use and installation.

Terms and conditions: Prices quoted are in U.S. Dollars and FOB Palo Alto, California. Freight allowed, 25 or more, anywhere within the continental United States.

Terms are net cash; check with order; or C.O.D. A 2% cash discount will be allowed on check with order.

A *Resale Certificate* should accompany dealer's initial order. California dealers not furnishing Resale Certificate must add 6% (or 6.5%) state sales tax.

Products may be mixed to achieve most favorable quantity pricing.

For direct export, please request our Export Schedule.

Prices and terms are subject to change without notice.

of your program. You'll know at once what simple variables and strings you've used and what values you've assigned to each.

```
FIND A$,100 – 200
  110 A$ = "HELD" + B$
  180 B$ = MID(A$,I,12)
  200 INPUT A$

READY.
```

FIND Like **LIST**, this command will show a set of lines. But the FIND command is followed by specifying a character string. Those lines, and only those lines, containing a desired character string will be listed on you PET's screen. If you were to type **FIND A$, 100-500** your PET's screen would display all lines between line numbers 100 and 500 that contain **A$**.

```
APPEND "INPUT"

PRESS PLAY ON TAPE #1
OK

SEARCHING FOR INPUT
FOUND INPUT
APPENDING

READY.
```

APPEND You've already worked hard to develop a number of programs and, of course, you've saved them on tape. You're working on a program that's now in memory. Just type in **APPEND** "*program name*" and all statements in that program will now follow the program in memory. No need to retype; no opportunity for errors. Save any set of statements or subroutines onto a tape, using the normal **SAVE** command; then recall them with the **APPEND** command and add them permanently to your program.

From an Expert:

"...my **Toolkit** *arrived...the ROM was thoughtfully placed in a conductive plastic pin carrier, protecting it from the possibility of bent pins and electrostatic damage.*

Also enclosed was a 34 page book of documentation. Professionally printed with a firm slick cover, it was a delight compared to some of the documentation that we have all seen in the mass micro market. I immediately got the impression that this company cared about what their customers thought about them. At this point I was favorably impressed and developed some confidence in the product and the company...

In my opinion the investment for your PET/CBM is more than worth it. The product performs great and with the guarantee...you can hardly go wrong."

...An author and frequent contributor to computing magazines

The Toolkit consists of two kilobytes of ROM firmware on a single chip.

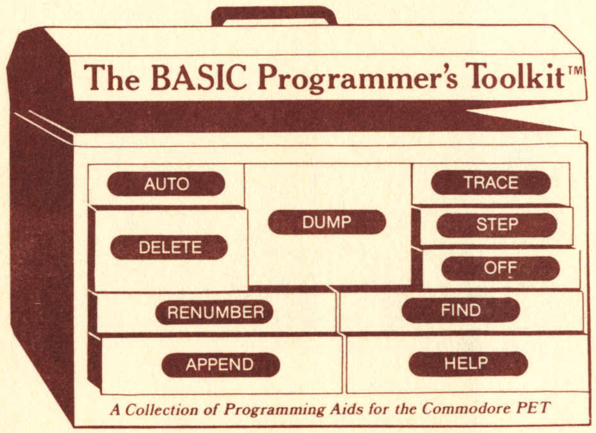
For the 16K and 32K PETs, the ROM chip plugs into a spare socket on the main circuit board inside your PET.

For the 8K PET, **the Toolkit** is mounted on a special printed circuit board with edge connector and attaches to the memory expansion port on the right side of the PET, and to the second cassette interface.

Memory addresses in ROM: $B000 to $B7FF; addresses in the PET RAM: $03E0 to $03FF

AVAILABLE AT:

Increase Your PET's IQ


The BASIC Programmer's Toolkit™
AUTO
DUMP
TRACE
DELETE
STEP
OFF
RENUMBER
FIND
APPEND
HELP
A Collection of Programming Aids for the Commodore PET

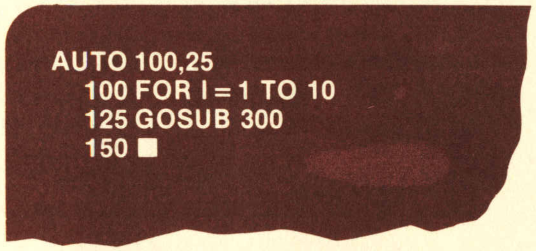
with the BASIC Programmer's ToolKit™

The Toolkit is a collection of machine language firmware aids designed to enhance the writing, debugging and polishing of BASIC programs for the PET. **The Toolkit** offers additional ROM storage, avoiding any need to load tapes or to give up valuable RAM storage.

The Toolkit *is fully assembled. It is not a kit and requires no special tools to install.*

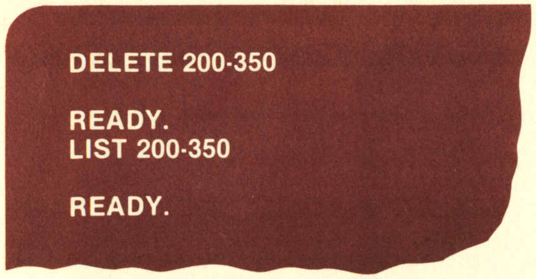
The Tools in the BASIC Programmer's Toolkit™

Power up your PET, execute the command **SYS** with the appropriate address and your *new* PET is off and running with these commands:



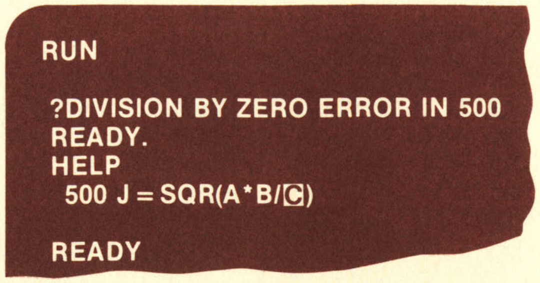
```
AUTO 100,25
  100 FOR I = 1 TO 10
  125 GOSUB 300
  150 ■
```

AUTO This command is followed by a series of optional parameters specifying where you want to enter lines and how far apart you want them. Your PET will automatically respond with a line number prompt. You won't have to enter the line numbers; you won't have to worry about errors of screen editing.



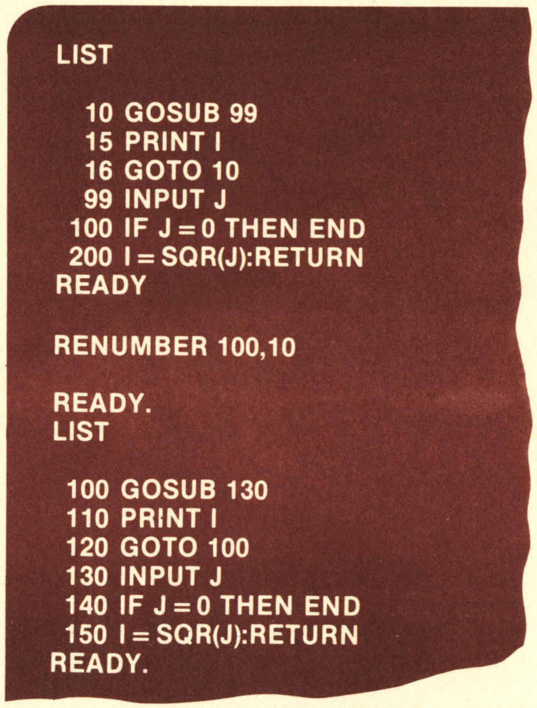
```
DELETE 200-350

READY.
LIST 200-350

READY.
```

DELETE Like **LIST**, this command is followed by a range of line numbers . . . and every one of the lines within the range of numbers will be removed instantly from your program. No longer will you have to type a line number, press **RETURN**, type the next line number, **RETURN**, next line. . . .



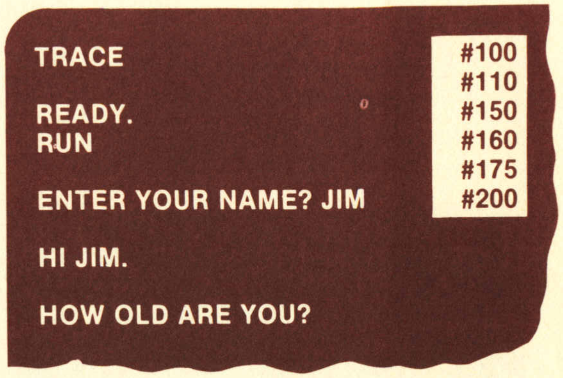
```
RUN

?DIVISION BY ZERO ERROR IN 500
READY.
HELP
 500 J = SQR(A*B/C)

READY
```

HELP How many times have you wanted to scream "HELP!" when your PET couldn't interpret your program and all it would say was: **?SYNTAX ERROR**? Fret no longer: Now just type in **HELP**. The line on which the error occurs will be shown and the erroneous portion of the line will be indicated in reverse video on the screen. Truly a great help in any learning or school situation.



```
LIST

   10 GOSUB 99
   15 PRINT I
   16 GOTO 10
   99 INPUT J
  100 IF J = 0 THEN END
  200 I = SQR(J):RETURN
READY

RENUMBER 100,10

READY.
LIST

  100 GOSUB 130
  110 PRINT I
  120 GOTO 100
  130 INPUT J
  140 IF J = 0 THEN END
  150 I = SQR(J):RETURN
READY.
```

RENUMBER Now you can change all line numbers— and all references to those numbers—instantly, as, for instance by evenly spaced increments of 100 or 25 or 10. . . .

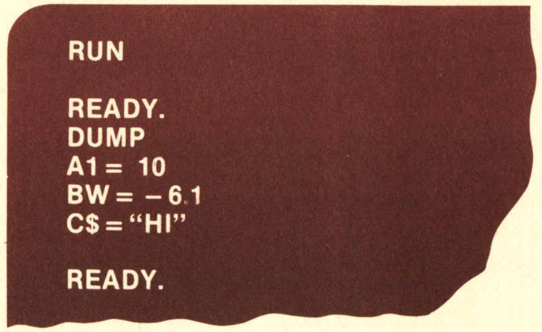


```
TRACE                               #100
                                    #110
READY.                              #150
RUN                                 #160
                                    #175
ENTER YOUR NAME? JIM                #200

HI JIM.

HOW OLD ARE YOU?
```

TRACE Now you can see precisely the order and sequence in which your program is being executed. You can also stop the program at any point and record the sequence. Type in this command and your PET will keep a record of the line numbers of the last six statements executed. These last six statement numbers will appear in a small rectangular window in the upper right hand corner of your screen.

STEP Again your line numbers are displayed in the upper right hand corner of the screen in this version of **TRACE**. But now your PET executes just one statement and pauses until you press **SHIFT**. Then it proceeds to the next statement.

OFF This command will stop either **TRACE** or **STEP**.



```
RUN

READY.
DUMP
A1 =  10
BW = – 6.1
C$ = "HI"

READY.
```

DUMP During or after running a program, this command will display the names and values of all variables used in the execution

THIS IS A TRIAL BALLOON

You'll want to see and sell the BASIC Programmer's Toolkit.™ So—as a trial offer—you can buy just one Toolkit for the original 8K PET and one Toolkit for the new 16/32K PET and get the ten-unit price. And, just this once, we'll even pay the freight. When you get the Toolkits, look them over, show them to your customers. If the Toolkits aren't at least as good as we say they are, return them to us in ten days for a full refund. Can anything be fairer? But, please remember, this is a one-time offer . . .*

FOR YOUR OPENING ORDER ONLY

. . . see how $89.18 puts you in business . . . ➽

**this offer good only until October 30, 1979*

PALO ALTO ICs • 430 Sherman Avenue, Palo Alto, California 94306

Yes, I want a Trial Balloon.

Please send me one each:

● The Toolkit Model TK80P complete with connectors to interface with an 8K PET. My special, one-time cost is **$56.00**

● The Toolkit Model TK160, ROM only, to interface with the 16K or 32K PET. My special, one-time cost is **$35.00**
And I understand that you'll pay all shipping and handling charges.

I enclose my check for $89.18. ($91.00 less 2% cash discount.)

I understand that if I'm not completely satisfied, I can return these in ten days for a complete and immediate refund.

My resale # is ______________________ for the state of ______________________

Company Name ____

Address ____

City ______________________ State ____________ Zip ____________

Phone number ____

Dealer principal ______________________ Title ____________

☐ I can't wait for the Trial Balloon; I want to get into business now. And I want to take advantage of the free freight and money-back guarantee. I'm enclosing my quantity order with payment method indicated.

A Sam Bernstein production. This went out to dealers on heavy ~~stock~~ yellow stock. (Sorry - we were out of originals).

PALO ALTO ICs A Division of Nestar Systems, Incorporated

430 SHERMAN AVENUE
PALO ALTO, CALIFORNIA 94306
415/327-0125

Office
of the
President

Dear PET Dealer:

Do you remember Marlon Brando, in "The Godfather," whispering hoarsely, *"I'm gonna make you an offer you can't refuse"*?

Well, I'm not the Godfather. . . I'm much nicer. . . but I'm going to make you an offer you can't refuse either:

The BASIC Programmer's Toolkit is guaranteed to bring new business into your store . . . new business that's profitable and trouble-free.

Buy just two of the new Toolkits: one for the original 8K PET, one for the new 16/32K PET. We'll bill you at the ten-unit price; we'll deduct 2% for cash; we'll ship freight paid.

. . .That's a profit of over $40.00 on an investment of barely $89.00.

Try the Toolkits, offer them to your customers, use the great point-of-purchase display. If, after ten days, you don't like the Toolkit for any reason whatsoever, return both Toolkits to us for full refund; no questions asked.

OK! But what is the BASIC Programmer's Toolkit? Like a New York Egg Cream Soda that's made with neither eggs nor cream, the Toolkit is not a kit and requires no tools.

The Toolkit is two kilobytes of ROM firmware on a single chip with extraordinarily powerful commands that will make programming a PET better, faster, easier. There's no need to load tapes or to give up valuable RAM storage.

Just listen to Greg Yob, a recognized expert, consultant and author in the world of BASIC programming:

"I'm a jaded expert in BASIC and I use more than half the Tools in the Toolkit constantly, the rest occasionally. It's got to be a necessity to anyone programming a PET. And more than worth the money.

"Anyone who's been using the PET for more than a few months and who's gotten past the PRINT statement must *need the Toolkit.*".

I remember the first response I received—the first of several hundred—when there were mentions of the Toolkit in *Cursor* and *Recreational Computing:*

> "*Your Toolkit description was like manna from heaven . . . please don't send me a card next month putting the shipping date off another two to six months or a year per [another manufacturer].*"

I'm telling him—and you—that we're not like [another manufacturer]. We've got them in stock right now. *Does that make us different?*

Or are we different when we tell you that we've got superb documentation right now, packed with every unit at no extra charge? A 38-page booklet that tells all.

Or are we different from [another manufacturer] when we say that we practice the best possible quality control . . . and put our money where our mouth is? The Toolkit is warranted for six months . . . **double** the normal warranty period. And if any Toolkit should prove defective within the warranty period, you can take it back from your customer, give him or her a new Toolkit, send the defective one back to us for full credit!

> *(As a matter of fact, you can even give your customer a ten-day free field trial. Now or at any time. If it's unsatisfactory, take it back and send it on to us for full refund or credit . . .* **even if it's not defective.***)*

And we're going to support the Toolkit in the right publications, complete with dealer listings including yours . . . if we have your permission.

☆ ☆ ☆

So there you have it. The $89.18 full-profit, no-hassle, new business. There's a convenient order form you can use. You'll make us both very happy.

I guarantee it.

Cordially.

Harry Saal

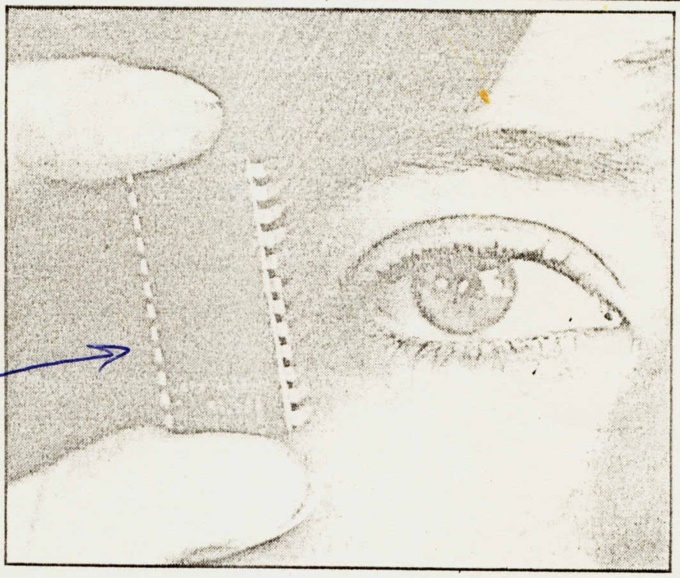
Harry Saal

PALO ALTO ICs A Division of Nestar Systems, Incorporated
430 SHERMAN AVENUE
PALO ALTO, CALIFORNIA 94306

Bulk Rate
U.S. Postage
PAID
Permit No. 408
Palo Alto, CA

DON'T YOU THINK YOU SHOULD KNOW THIS NEW MONEY-MAKING BREAKTHROUGH BEFORE YOUR CUSTOMER FINDS OUT AND TELLS YOU?

An ad placed by our English distributor.



That's an IC, by the way.

EYE SEE, IC

Petsoft are starting to make deliveries of the Toolkit which we reviewed a month or so ago. The delay is simply due to the fact that so many of you have ordered them, supply and demand equals overload. Shipments are getting through so please don't fret. For those of you who are recent converts the Toolkit is a 2K ROM package that plugs directly into a 16 or 32K PET and gives you a range of editing and debugging tools directly accessible through BASIC. For those of you with the old machines the ROM plugs onto the expansion port on a little PCB. The cost is £55 for the ROM or £75 for the plug-on version. The product is also Commodore approved, it certainly got our commendation as being a very useful piece of kit. For more info contact Petsoft at PO Box 9, Newbury, Berkshire.

An ad in the Japanese computer magazine

ベーシック・プログラマーズ・ツールキット

Commodore PET用 BASIC PROGRAMMER'S TOOLKIT™

Basic Programmer's Toolkit™はコモドール・PETのプログラムの開発・デバッグを容易にし、より完成度の高いものにするためのファームウェアです。

Toolkitは2KバイトのROMに収納されており、カセットテープは不要で、RAMも占有されません。

Basic Programmer's Toolkit により以下の新しいコマンドが使えるようになります。

AUTO	自動文番号発生モードに入り適当な間隔をあけた文番号がPETにより作られる。
APPEND	RAM上のプログラムとカセットテープのプログラムとを結合する。
DELETE	ＬＩＳＴと同様な使い方で、ある範囲の文を削除する。
DUMP	シンボルテーブル中の変数名とその値とを表示する。プログラム実行前でも実行後でも使用できる。
HELP	ＢＡＳＩＣでエラーがでた場合、そのラインをディスプレイし、さらに問題の箇所をハイライト表示する。
RENUMBER	ＢＡＳＩＣプログラムの文番号を特定のステップで付けなおす。
TRACE	実行されている文番号をスクリーンのコーナーに表示する。ワンステップづつの実行もできる。

Basic Programmer's Toolkit™は現在デモンストレーション可能で、量産型は（新型のＰＥＴでも旧型でも使えます…ＰＥＴ2000－8、16、32）これらには説明書と使用例が含まれます。

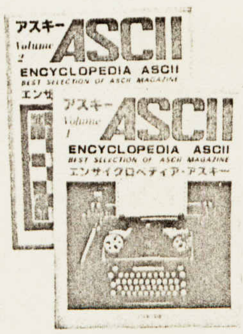
10月発売予定，お申し込みは使用機種（8，16，32型）を必ず記入してください。

¥29,000　〒500

●お申し込みは現金書留で、Ｂ係までお願い致します。

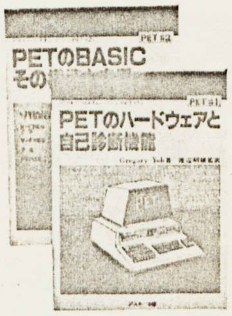
●〒107東京都 港区南青山5-6-4
ハイトリオ　☎(03)407-4910

株式会社 アスキー出版



●エンサイクロペディア・アスキー
▦合本1　77年7 月号～78年2月号
▦合本2　78年3月号～78年8月号
この合本は掲載されている各号のアスキーの記事をすべて検討し，再構成して刊行しています。発売後に変更点，問題点の発見されましたリスト・図などは可能な限り訂正されています。

各3500＋送料200円

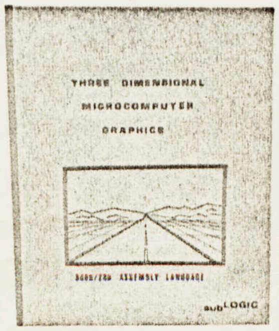


●アスキー・システム・バンク
▦PETのハードウェアと自己診断機能
PETの置き方持ち運び方にはじまり，セカンドキーボード，オリジナル・キャラゼネなどの各種改造法，自己診断ルーチンの走らせ方，ROM，RAMチェックプログラムなどが詳説されています。PET（8,4型）の全回路図掲載。
▦PETのBASIC・その構造と応用
追加ロード（アペンド）機能や，実行中に自分自身を変更するプログラムなどを例にとり，実際に読者とともに作っていきます。位精度のグラフィックスを用いたりサージュ波形表示プログラムなど，多彩な内容は第3巻に続きます。

各2500円＋送料200円

●Basic Computer Games
マイクロコンピュータ・ゲーム集
単純な数あてゲームばかりの本とは一味ちがいます。CAIや"発見的解法"，3次元グラフィックスなど，高度なアルゴリズムも多数含まれています。

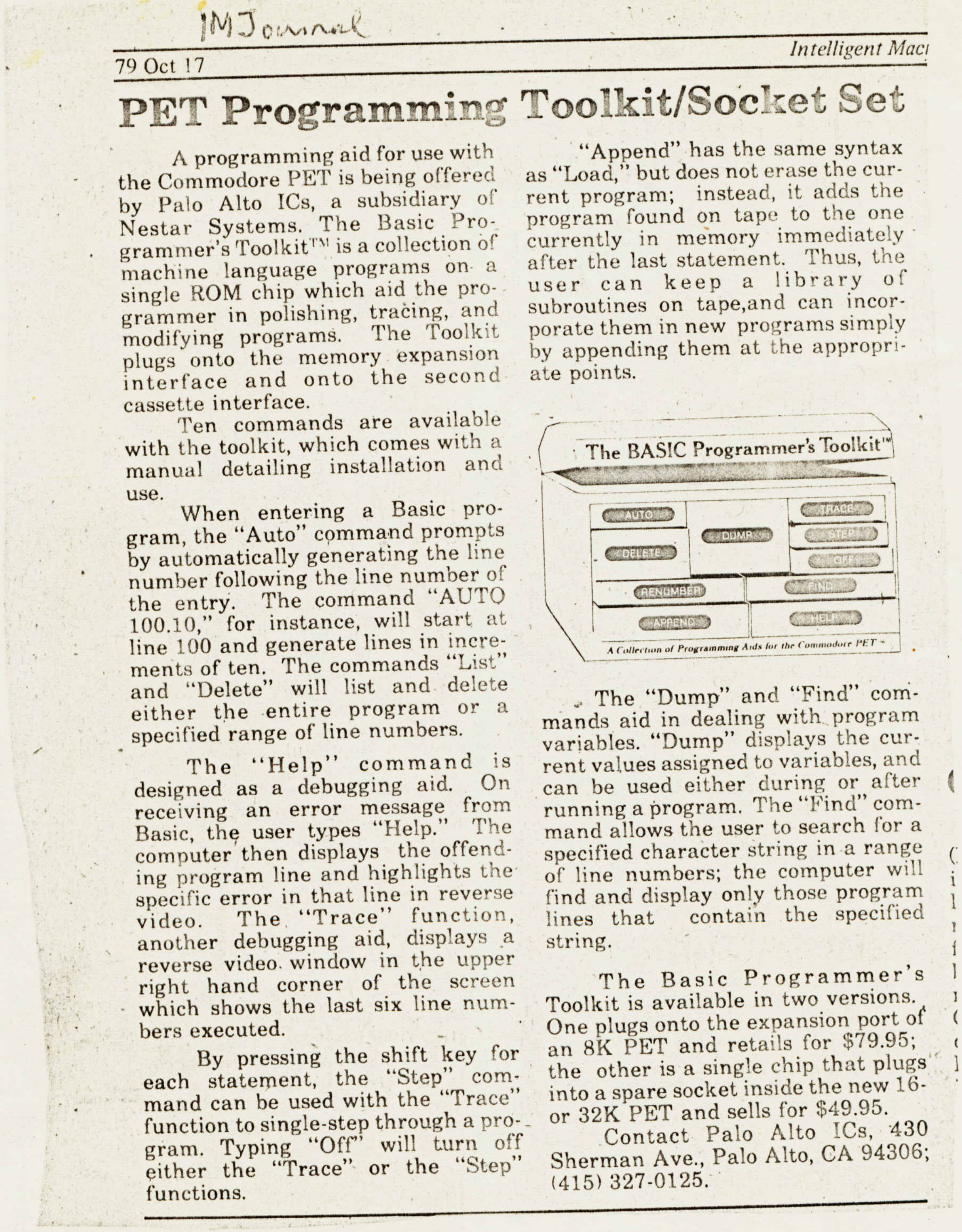
2900円＋送料100円　　日本語版



●Sub LOGIC GRAPHICS ONE
3次元のグラフィクス
256×256や128×128表示のできるグラフィクスディスプレイが手にとどくようになってきました。このソフトウエアではそれを可能にしています。その他3次元のデータを2次元の画面のデータに変換も可能です。

BASIC	12000円＋送料200円
8080/Z80	16000円＋送料250円 ターベル規格テープ
6800	15000円＋送料250円 カンザスシティ規格テープ

A typical product announcement for the Toolkit.



IMJournal

PET Programming Toolkit/Socket Set

A programming aid for use with the Commodore PET is being offered by Palo Alto ICs, a subsidiary of Nestar Systems. The Basic Programmer's Toolkit™ is a collection of machine language programs on a single ROM chip which aid the programmer in polishing, tracing, and modifying programs. The Toolkit plugs onto the memory expansion interface and onto the second cassette interface.

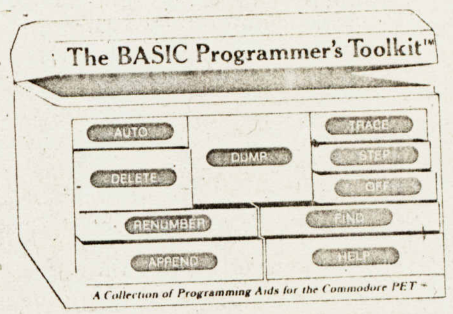
Ten commands are available with the toolkit, which comes with a manual detailing installation and use.

When entering a Basic program, the "Auto" command prompts by automatically generating the line number following the line number of the entry. The command "AUTO 100.10," for instance, will start at line 100 and generate lines in increments of ten. The commands "List" and "Delete" will list and delete either the entire program or a specified range of line numbers.

The "Help" command is designed as a debugging aid. On receiving an error message from Basic, the user types "Help." The computer then displays the offending program line and highlights the specific error in that line in reverse video. The "Trace" function, another debugging aid, displays a reverse video window in the upper right hand corner of the screen which shows the last six line numbers executed.

By pressing the shift key for each statement, the "Step" command can be used with the "Trace" function to single-step through a program. Typing "Off" will turn off either the "Trace" or the "Step" functions.

"Append" has the same syntax as "Load," but does not erase the current program; instead, it adds the program found on tape to the one currently in memory immediately after the last statement. Thus, the user can keep a library of subroutines on tape, and can incorporate them in new programs simply by appending them at the appropriate points.


The BASIC Programmer's Toolkit™
AUTO
DUMP
TRACE
STEP
DELETE
OFF
RENUMBER
FIND
APPEND
HELP
A Collection of Programming Aids for the Commodore PET

The "Dump" and "Find" commands aid in dealing with program variables. "Dump" displays the current values assigned to variables, and can be used either during or after running a program. The "Find" command allows the user to search for a specified character string in a range of line numbers; the computer will find and display only those program lines that contain the specified string.

The Basic Programmer's Toolkit is available in two versions. One plugs onto the expansion port of an 8K PET and retails for $79.95; the other is a single chip that plugs into a spare socket inside the new 16- or 32K PET and sells for $49.95.

Contact Palo Alto ICs, 430 Sherman Ave., Palo Alto, CA 94306; (415) 327-0125.

Appendix 4

BASIC Programmer’s Toolkit reviews

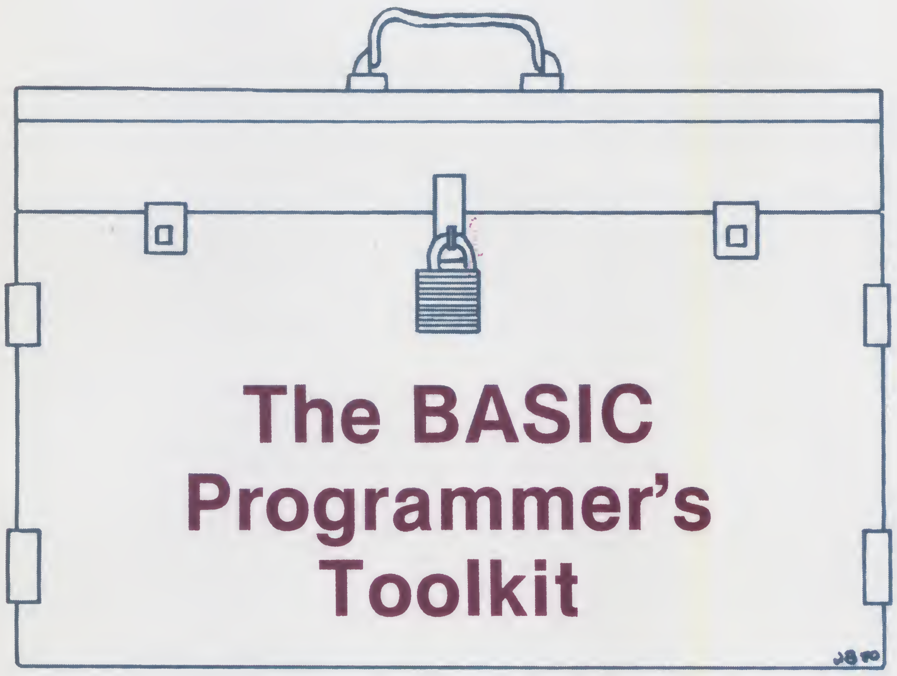
April 1980
$2.50

kilobaud

MICROCOMPUTING T.M.

for business . . . education . . . FUN!

Tom Hayek
723 Hialeah Dr.
Racine WI 53402



The BASIC Programmer's Toolkit

Lift the lid on Nestar's toolkit and have a look inside.

In August 1979 I first noticed advertising descriptions of the BASIC Programmer's Toolkit, which claimed to add ten more commands to the Commodore CBM BASIC vocabulary. The commands are: AUTO, DELETE, RENUMBER, APPEND, DUMP, TRACE, STEP, OFF, FIND and HELP. Since the commands are in firmware (ROM), they will not take up any of my 32K of RAM; they will be there whenever I turn on the computer.

I was eager to have this additional command capability and placed my $49.95 order for the ROM, which is a product of Palo Alto ICs, a division of Nestar Systems, Inc.

After a short delay, my Toolkit arrived by first-class mail. I eagerly opened the padded mailer to find the ROM placed in a conductive plastic pin carrier, protecting it from the possibility of bent pins and electrostatic damage. Also enclosed was a 34-page book of documentation, professionally printed with a firm, slick cover.

I immediately got the impression that this company cared about what their customers thought about them. At this point I was favorably impressed and developed some confidence in the product and the company.

Installation

The documentation begins with a brief description of the commands and an explanation of the Toolkit installation, followed by a clear and complete description of each command, including examples. Installation in my 32K CBM with a full-size keyboard is a simple matter of opening the case and installing the 2K ROM in one of the three existing empty sockets. Installation on an 8K PET involves plugging a small board containing the ROM and some address decoding into the memory expansion port on the side of the PET. Another small connector with a single wire plugs into the second cassette port to supply 5 V dc to the board.

As the Toolkit manual warns, turn off the power and disconnect the computer from the ac line. Ground yourself by touching the metal case of the computer to dissipate any static charge just prior to handling the Toolkit for installation.

The ROM occupies memory positions B000-B7FF (hex), so if you have other memory expansion systems, such as Skyles or ExpandaPet, be sure to find out—from their respective documentations—the correct socket to plug into. Also, be sure to have the correct orientation of the ROM with respect to pin one.

I installed the ROM and checked for any bent pins. I was now ready to check it out. After powering up my CBM, I entered a BASIC command SYS 45056. This initializes the Toolkit ROM, and the CRT should read: (C) 1979 PAICS. Everything went smoothly, and I was ready to explore all the BASIC commands now available from the Toolkit in the direct mode.

The Commands

AUTO—provides automatic line numbering as you type in your BASIC program. The general syntax is:

AUTO beginning line number, interval.

If you type in AUTO without specifying any parameters, line numbering will start with 100 and the interval will be 10. To get out of the AUTO mode, just hit the return key without entering anything after the line number.

AUTO also remembers where it left off. If you exit the AUTO mode to do some editing and then type AUTO, numbering will start at the next sequential line in your program. The previously set interval will be maintained. If you type AUTO 200, line numbers will start with 200 and be incremented by the last interval given in the AUTO command.

AUTO helps to ease some of the typing drudgery in entering a BASIC program.

RENUMBER—renumbers the entire BASIC program presently in memory. All GOTO, ON . . . GOTO, GOSUB, ON . . . GOSUB, IF-THEN, RUN and LIST commands are also changed to the new respective reference line. All references to nonexistent line numbers are changed to 63999. This is especially useful when used with the FIND command. The general syntax is:

RENUMBER beginning line number, interval.

If you type RENUMBER without specifying any parameters, renumbering will start with line 100 and the interval will be 10. It took about 30 seconds to renumber a 10K program.

DELETE—removes BASIC lines by specifying the line number or range of line numbers in the same way that the PET/CBM LIST command lists lines. For example, DELETE 50 deletes line 50; DELETE 50-100 deletes lines 50 through 100; DELETE -100 deletes all lines from lowest through 100; and DELETE 100- deletes all lines from 100 through highest.

The Toolkit is designed so that if you type DELETE without giving a range or specific line number you will get SYNTAX ERROR?. This prevents the loss of the entire program by mistake.

APPEND—will load a program from a cassette and add it to the end of a program already in RAM. It works in the same way as the PET/CBM BASIC command LOAD. The general syntax is:
APPEND "program name," cassette drive (1 or 2).

As with the PET/CBM LOAD command, no specification of the cassette drive defaults to cassette drive #1.

APPEND is convenient for adding previously written subroutines to a program in RAM. You could have several often-used subroutines stored on tape and APPEND them to an existing program under development in RAM.

Caution: You must keep the line numbers in order. APPEND will add anything on the tape to the end of the program in the computer. It is a good idea to number all of your subroutines in the 60000-63000 range and not use this range for your BASIC main body programs. This will help to avoid conflicts in duplicate line numbers when appending.

FIND—locates and displays all lines that contain a specified BASIC keyword, section of a BASIC statement or a quoted string constant. The general syntax is:
FIND BASIC code, line number-line number
FIND "string", line number-line number.

The line-number-parameter-search range performs the same as the PET/CBM LIST command range and the Toolkit DELETE command range. If you omit the line number parameters, the whole program will be searched.

FIND allows you to be as specific as necessary when detailing the BASIC statement or string parameters. For example, FIND FOR I will locate and list every line containing FOR I; FIND A will locate and list every line *containing* the variable A; FIND "THIS" will locate and list every line containing the word THIS; FIND GOTO 100, 10-20 will search lines 10 through 20 and list all lines containing GOTO 100.

As you can see, this proves to be a valuable time-saver. Recall in the description of RENUMBER that any references to nonexistent line numbers are assigned a value of 63999. Now we can use the statement FIND 63999 to list any bad references in the program.

When you use FIND, the number of lines listed on the CRT may be sufficient to cause scrolling. You may slow down the scrolling by holding down the RVS key or stop it anywhere with the STOP key.

DUMP—displays all the non-array variables in memory. They are displayed in the form: variable name = present value (i.e., A = 2). This is a great help in debugging programs. Putting STOP statements in the program and then checking the variables at that point is one way to find out where the program is amiss.

DUMP may fill the CRT and cause scrolling. This can be stopped by holding down the SHIFT key. Releasing the SHIFT will allow the scrolling to continue. The STOP will cause the scrolling to stop and abort, as in FIND.

HELP—When you encounter an error while running a program, the PET/CBM will stop the program and print an error message and line number. The HELP command will list the line and indicate the error within the line with a reverse field cursor. The syntax is: HELP.

HELP must be executed before anything else after an error message, otherwise the source of the error will be lost. In that case, executing HELP will do nothing and the computer will come back with READY.

The cursor is usually placed on the character just before the error, but in some cases will be on the error. In the case of 10 B = A / 0, the cursor would be on the 0 (division by 0 is an error).

TRACE—turns on a tracer mode, which will display the currently executed line number when the program is running. The last six line numbers are visible in a reverse field window printed in the upper right-hand corner of the CRT. These six lines scroll from bottom to top within the window, with the most recent line number at the bottom.

Pressing SHIFT will slow the program and scrolling down to about two lines per second.

STEP—also activates the tracer mode, executing one line of BASIC at a time. The line numbers and reverse field window appear just as in TRACE. To execute the next line, momentarily press SHIFT. If you hold the SHIFT key down, the program will continue to run until SHIFT is released. To stop, simply press STOP.

STEP can be conveniently used in debugging also. Being able to single step through a suspected problem area aids in locating the possible faulty coding.

OFF—turns off either the TRACE or STEP commands.

Types of Toolkits

There are basically two types of Toolkits: a 2K ROM that plugs into an empty socket in the new PET/CBM (16K/32K) or an expansion board such as BETSI, and the ROM and an interface IC mounted on a small PC board that plugs into the memory expansion port on the 8K PET. This board has a single wire with a small connector that plugs into the second cassette port to supply 5 V dc for the board.

The costs of the two types are $50 and $80, respectively. The Toolkit comes with a money-back guarantee if you are not completely satisfied; there is also an exchange policy. If you purchase a Toolkit for a PET with the old ROM set and then decide to update to the new ROM set, you can exchange your Toolkit for one that will work with the new ROM set for $15.

Conclusion

Palo Alto ICs and Nestar Systems are not mail-order houses. Do not try to order from them, as you will only delay in getting your Toolkit. You should order from your local computer store. The only mail-order firm that I have seen advertising the Toolkit is Skyles Electric Works, 10301 Stonydale Dr., Cupertino CA 95014. ■

INFOTREE

THE FIRST PERSONAL INFORMATION MANAGER

WRITTEN IN

APPLE PASCAL

FEATURES

Keyed to Access to Data • Recursive Multi-Level Key to Access to Segment Data in any way YOU Want • Auto Command Type-In and Command/Key Recall • Displays or Prints Data for Single or Multiple Related Keys • Mailing Labels • 4380 Records Per Disk • Average Acesss/Update Time 10 Seconds.

APPLICATIONS

Mailing Lists • Phone No. Directory • Memos • Appointments Scheduling • MUCH MORE.

The Complete Package, including Manual is only $39.95. Get a Demo from your dealer, or write

Siro-tech ✓172

6 Main Street
Ogdensburg, N.Y. 13669
(315) 393-2640

Coming Soon: Mail-List • Text Pro • Auto Letter • Inventory • Point-of Sale • All Modular. All Interfaceable. All in Pascal

An article in a British magazine

The ten commands: will they change your life?

There's no need to step through fire to bring the tablets down from Sinai. John James takes a sceptical look at ten (or is it nine?) new add-on machine-code subroutines for the PET, holds them nervously in the palm of his hand, and concludes they may change your life, too.

WHAT EXACTLY is the Basic Programmer's Toolkit? Well, answering physically, it'll depend on whether you have an 'old' or 'new' PET. And how do you tell? If you have one of the small keyboards, then for our purposes you have an 'old' PET; a big keyboard (and therefore a separate tape cassette unit), and you have a 'new' PET.

The only effect this has on the Toolkit is on the type of Toolkit you get.

For an 'old' PET, you get a small printed circuit board, with a chip and some other bits mounted on it, and a connector loose-wired to its edge. The board has another connector mounted directly on it, which lets it be plugged straight into the memory expansion board port on the right-hand side of your PET. The loose-wired connector is then plugged into the second cassette port, just round the corner, at the back.

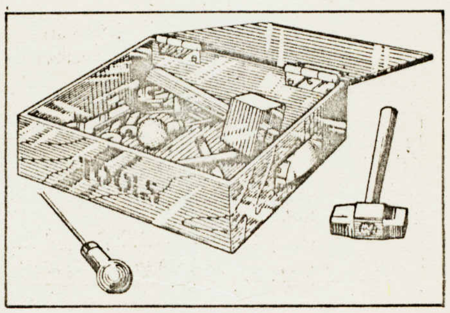
For a 'new' PET, you simply get a chip. This you take in nervous fingers (try holding a chip that cost over sixty quid, and see if you aren't nervous), to plug straight into an empty socket on the main board inside your PET.

Obviously, installation on an 'old' PET couldn't be easier, so there's no point in covering the question 'can you do it simply?'. Fitting a 'new' PET sounds a bit more tricky, so how hard is it?

Nothing could be more straightforward, thanks to one of the best instruction books I've seen in many a day. If you follow the book, you really can't go wrong. It's worth saying here, for the knowledgeable, that the book also covers all the things you'd expect, like avoiding static discharge, bent pins, and so on. In other words, fitting should be no problem at all — except if you've had expansion memory fitted to your PET. Here, the book recommends that you 'contact your dealer for installation information'.

Now there is room for criticism here, since you won't be reading the book until after you've got your Toolkit. Have you ever bought a super-gadget-gizmo, hurried home in high excitement, torn the wrapper off, and then found you hadn't got the necessary battery because nobody said you'd need one? Familiar with the frustration this causes?

Then be warned. If you have got expansion memory, tell your dealer at the



time of buying, and make sure that he can advise you.

Once you've fitted the Basic Programmer's Toolkit, what will it do? Here, I'd like to come back briefly to the instruction book that comes with the Toolkit. It's 34 pages long, nicely printed, totally explicit, clear as crystal, loaded with example programs, and best of all, originally written by Gregory Yob.

The cognoscenti will have twitched at that magic name; others should know that Gregory has not only been in personal computing since Heaven knows when — to many, he *is* personal computing.

This means that you'll get the best out of the Toolkit right from the word 'go', and the best can meant quite a lot. Yet there is room for another criticism. The Toolkit is widely advertised as adding '10 powerful new commands to PET's Basic', which I think is not entirely fair.

In fact, you get nine new commands (we'll have a look at how powerful they are in a moment); the tenth isn't really a command, in my view, since it simply switches off a couple of the other nine.

So what are these nine powerful commands? The best way of answering that is to look at them one by one, but before we do that, let's return to that magic moment just after you've fitted Toolkit, and you're about to start trying it out.

You have to turn it on, and this you do very simply by entering SYS 45056. PET should respond immediately by displaying (C) 1979 PAICS.

From that point on, all nine active commands are available until you switch PET off. You get them back, after powering down, by entering the SYS instruction again.

Reviewing the nine commands gives me a choice: I can tell you about them alphabetically, or in order of personal favouritism, and I'm going to opt for the latter.

Let's start with RENUMBER.

Now this one is really good (not that the others aren't, but I said I'd start with my favourite!). The major point to make immediately is that RENUMBER is fully professional.

In other words, forget about those previous methods, using add-on subroutines, which required you to go right through the program, looking for line numbers, to make sure each one had enough space before it for whatever it might become when renumbered.

The Toolkit RENUMBER expands or contracts line numbers, and closes everything else on the program line up tight to the new number, whatever it might turn out to be.

You activate RENUMBER by typing the word and entering it. If you do this, Toolkit assumes you want to start at Line 100 and go in steps of 10.

If you want to start at any other line number, and/or go in other increments, you simply say so and it happens — and pretty fast too. I linked four programs together, all about 6K long, and renumbered the lot in less than 15 seconds.

Of course, it depends on the number of renumbers (if you get my meaning), but you shouldn't have any complaints about the time you hang about.

There are two other goodies in RENUMBER.

First, if renumbering will take your line numbers over the maximum permitted, the routine aborts, and no renumbering takes place. You'll like that if you've ever 'crashed' with other methods!

Second, if the routine comes across a line number that doesn't exist later in the program (as in GOTO X, and X isn't there, for instance) then it quite deliberately renumbers that to 63999.

This ties in nicely with my next most-liked command, FIND.

FIND locates all lines that contain absolutely anything you like to specify. It's important to be clear about this, because the advertising for Toolkit isn't. The advertisements say that FIND locates lines 'containing a desired character string'. That's quite true, but it's misleading, in that you might well think that all you can ask FIND to do is look for strings.

Not so. FIND will do much more than that (which certainly means the advertisements err on the right side!). FIND will find anything. If you want to know which lines have the variable 'G' in them, and you enter FING G, then every line with the variable in it will be found and displayed.

Note that FIND G won't find GOTO or GOSUB, just because they have a 'G' in them, which saves a lot of confusion. To find GOTO statements, you need to enter FIND GOTO, and the same holds good for any other Basic statement. If, on the other hand, you enter FIND "G", then every quoted string that contains the letter G, whether it's on its own or contained in a word, will be found and displayed, which is not only useful, but fun.

Now you'll see the point of renumbering non-existent lines to 63999. All you need to do to find them is enter FIND 63999, and presto! there they all are, displayed for your pleasure. And, what's more, you can then go right ahead and edit them on screen, which helps resolve little local difficulties quite quickly.

Keeps on adding

I think my third favourite must be APPEND. This one adds program to program to program, for just as long as you have memory available to keep doing it. You only have to keep one thing in mind: APPEND does not replace program lines in memory with new similarly-numbered lines from the material you're appending, not does it interleave program lines. Quite simply, it does exactly what it says: it keeps on adding program lines to the end of whatever might already be in memory.

It does this from tape only, however, and will not append from disk or other devices. I don't see this as a very great disadvantage, but perhaps some people will.

All I can say is that I've already found the command incredibly useful, in only a few days of playing about. If I have a favourite subroutine or whatever on disk, I don't really feel it's a great hardship to off-load on to cassette so that I can use the APPEND facility.

My fourth favourite? Well, we're getting to the photo-finish stage now, but maybe a whisker or so in front is HELP. You use this when that lovely point comes, as you finish typing the world's finest program, which of course is totally error-free, and run it, when everything judders to a standstill halfway through, with PET beadily blinking ERROR at you.

You look at the line allegedly in error, and it seems perfect, just like the rest of the program.

Fret not; if you'd like to know where that little but is lurking, enter HELP. Up comes the line again, but this time a bit of it will be in reverse field. This, says the manual, is where the error is. Well, I found this wasn't strictly true, but it was close enough to make no difference. Sometimes the reverse field is to the left or right of the error, sometimes a character or so distant, but it's never far away. After all, if the error was something missing, it'd be hard to put a reverse field on what wasn't there!

Mounting panic

The fifth and sixth commands are similar: TRACE and STEP. Both put a little reverse-field window on the top right-hand corner of the screen, which displays up to six line numbers. With TRACE, the program runs steadily, and the line numbers being executed whip up the litttle window at a high rate of knots.

The manual says that TRACE slows the running of a program considerably, and that pressing the shift key slows it still more, to around two lines per second. Even at that, though, it's still moving fast, and I haven't so far found TRACE overpoweringly useful. The only effect it's had on me to date is to induce a mounting panic as I tried to keep up with what was happening. These are early days, though.

STEP seems rather more sensible, but to be harsh, it's merely an extension of TRACE, in that it displays one line number only, and that's the one that's just been executed.

For debugging, my vote goes to STEP, however. Not that my programs ever have bugs in them, of course, but I can always offer my help to those less fortunate!

One smart thing about both TRACE and STEP: the reverse-field window overrides any screen display, which avoids peering at line numbers through overwritten screen characters. This is a disadvantage too though, because it's possible (but, to be fair, odds-on unlikely) that a bug you're trying to spot will be right under that little window!

DUMP comes next, and I have a feeling that I'm going to love it more and more. You use it most often at the end of a program, and entering it causes every single variable (excepting arrays) to be printed, with the value each has at that time.

Now I'm a messy programmer: in other words, I like to program straight on to the screen. Not for me the clean, aseptic approach of writing everything out beforehand.

Thus I often get to a point where I'm uncertain which variables I've used, which I haven't, and what values I've given those that I have put in. I'm sure that you, gentle reader, never do this, but just in case there's someone out there who does, DUMP could be exactly what he or she needs.

Family resemblance

The final two commands — AUTO and DELETE — have a family resemblance also. Their names explain virtually all. AUTO provides automatic line numbering, in whatever increments your heart desires, as you type furiously through your program-entering process. DELETE wipes whole blocks of lines out, between whatever numbers you specify.

That, at least must save wear and tear on the Return key but, ironically enough, the other commands available in Toolkit tend to lessen the usefulness of DELETE for me. I'd have loved it in those not-so-far-off days when I used subroutines tacked temporarily on to the front or rear of the main program.

And there it is: the Basic Programmer's Toolkit, with 2K of ROM firmware, consisting of a collection of machine language routines, all on one chip, adding nine (or 10, if you want to be finicky) new commands to your PET.

Is it worth £60? I would say yes.

Petsoft tell us:

- They are writing a British annexe to the Minimax manual with the intention of 'making a number of contributions aimed at the naive user.'
- This business software, available for the PET on Computhink mini-floppies, has been adapted and integrated for the Minimax. This means that programs from the suite can be run without having to 'bust through a series of complicated disc-to-disc steps.'
- Software titles which will be available early in the new year are Sales Accounting and Invoicing, Purchase Accounting, Stock Control, Word Processing and before the new financial year, Payroll.

Petsoft are also publishing a range of 50 utilities, languages and simple business routines. During the course of the year Petsoft will release some graphics packages supporting animated graphics, 3D plotting and a "small-talk" type package.

The CPU, in additiion to recognising all standard 6502 instructions, will also support 64 user programmable op codes. These can be used for Pascal and Forth operations.

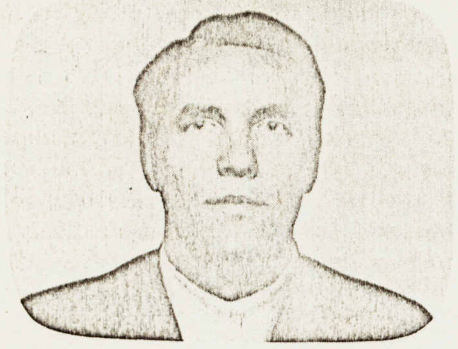
CompuThink themselves have Pascal, a compiled Basic and Fortran under development for release early in 1980.

Conclusions

- **It's added a lot more fun to my programming**
- **It's made bugs much easier to find**
- **It's encouraged me to be a little more adventurous in trying out different methods, because I know, if they're not right, I'll be able to locate them and change them**
- **Maybe best of all, it has let me go back to half-finished programs from long ago, and tackle them again.**
- **Toolkit also makes conversion of old-ROM programs very much more simple.**
- **If you're any sort of 'hacker', what are you waiting for?**

a 3-page article in a german magazine

GEWUSST WIE!



Erweitertes Basic für PET

Pierre J. KEMMLER HH F

Dieser Beitrag beschreibt einen preiswerten Hardwarezusatz, der beim Commodore-PET zehn zusätzliche Befehle ermöglicht, welche vorallem beim Entwickeln von Basic-Programmen eine wertvolle Hilfe sind.

Soeben traf aus Kalifornien ein neuartiger 'Werkzeugsatz' für PET-Programmierer ein, dessen englischer Name "BASIC-Programmer's Toolkit" lautet und der über erstaunlich klein ist. Es handelt sich hier um ein zusätzliches ROM-Modul, also um sogenannte Firmware, welche einem solchermassen erweiterten PET zehn neue, zum Teil langersehnte Befehle erlaubt:

- APPEND
- AUTO
- DELETE
- DUMP
- FIND
- HELP
- RENUMBER
- STEP
- TRACE
- OFF

Diese 10 neuen Befehle sind speziell bei der Erstellung und 'Entwanzung' (Debugging) der BASIC-Programme wertvoll. Wir kommen weiter unten im Detail auf diese Befehle zurück. Sie sind Subroutinen in Maschinencode, die in einem 2K Byte-ROM residieren, das in 2 oder 3 verschiedenen Ausführungen erhältlich ist. Eine Version ist für den PET mit neuem Operations-System und zwei weitere Versionen für PET mit dem alten Operations-System. Die zwei Versionen für alte PETs kommen daher, dass normalerweise bei diesen Geräten das Toolkit an den Memory-Expansion angeschlossen wird. Dazu wird das "Toolkit" komplett mit einer kleinen, doppelkaschierten Platine geliefert, die dort angeschlossen wird. Ein zweiter über eine flexible Leitung mit der nötigen Speisung vom Anschluss des 2. Kassettenrekorders her. Zudem befindet sich auf der Zusatz-Platine ein zweiter, freier Sockel für eine allfällige ROM-Erweiterung dessen Adressen-Leitungen bereits decodiert sind. Ist nun aber ein alter PET in seinem Speicherbereich erweitert worden, so ist ja der Memory-Expansion-Anschluss bereits belegt. Sollten Sie über ein Expandamem-System verfügen, so kann das Toolkit unter Verwendung einer entsprechenden Platine in eine der freien Kontaktleisten dieses Systems eingesteckt werden. Bei den neuen PET wird das Toolkit lediglich in einen der freien ROM-Sockel gesteckt. Wie Sie sehen, ist der Einbau des Toolkit in jedem Fall sehr einfach.

Doch jetzt wollen wir im Detail zu den Vor- und Nachteilen dieses Toolkits kommen. Zuerst die Nachteile:

- Wenn Sie über ein Compu/Think-Floppy-System verfügen, so müssen Sie sich zwischen dessen DOS (Disk-Operating-System) und dem Toolkit entscheiden! Beide können nämlich nicht gleichzeitig im PET residieren, denn beide benützen denselben Speicherbereich (Dezimal 45056, respektive Hex BØØØ aufwärts).

- Um Variablen abzuspeichern, benützt das Toolkit den oberen Teil des 2. Kassettenbuffers, so dass Programme, welche diesen Bereich ebenfalls benützen, zusammen mit tionen hervorrufen können. Allerdings ist diese Doppelbelegung beim Befehl APPEND vermieden worden.

- Drei kleinere Nachteile sind, dass die 10 neuen BASIC-Befehle nur im Direct-Command-Mode über die Tastatur, also nicht durch ein Programm aufgerufen werden können. Zudem lassen sich die Toolkit-Befehle nur allein, das heisst ohne Kombination mit anderen BASIC-Befehlen in derselben Zeile ausführen. Da das Toolkit nach dem Einschalten des PET mit einem SYS-Befehl initialisiert werden muss, hat dies nach jedem Reset erneut zu geschehen.

Doch nun zu den überwiegenden Vorteilen:

- Sehr leichter Einbau; sozusagen ohne Werkzeuge möglich.

- Vernünftiges Preis/Leistungsverhältnis.

- Dadurch, dass die 10 Befehle in einem ROM gespeichert sind, wird kein RAM-Raum zu deren Abspeicherung benötigt (ausgenommen die wenigen Bytes im 2. Kassetten-Buffer). Damit steht der vorhandene RAM-Bereich voll zur Verfügung, was speziell beim 8K PET von Vorteil ist.

- Alle 10 neue Befehle lassen sich analog zu den bisherigen BASIC-Befehlen des PETs abkürzen, indem lediglich der erste Buchstabe und

<u>Appendix 5</u>

draft press release on
million-dollar sales and counterfeiting

1-1-1

Hjs: of staff: No of 2/14/80

– LJS suggestions 2/17/80

Nestar Systems Inc.
NE 020
Feb. 14, 1980

V2

for more information, contact:
Ellen V.B. Lapham (415) 494-6267
Harry J. Saal (415) 327-0125

FOR IMMEDIATE RELEASE

NESTAR SYSTEMS ANNOUNCES A MILLION DOLLAR INTERNATIONAL WINNER

Palo Alto, CA. Feb. 14, 1980. Nestar Systems Incorporated, a Palo Alto based manufacturer of products for popular microcomputers, today announced that its BASIC Programmer's ToolKit^tm , a ROM (read-only memory) plug-in for the Commodore PET, has passed a million dollars in retail sales. Providing ten programming aids, such as 'append', 'renumber', and 'find', the BASIC Programmer's ToolKit^tm has gained wide success in Europe and North America.

l.c.

l.c.

The occasion was marred, however, by the news that unauthorized copies of the ROM-based ToolKit are being sold ~~in~~ on tapes ~~or~~ and diskettes ~~form:~~ ~~the change of media, from ROM hardware to magnetic media, does not alter~~ ~~the~~ this raises legal and ethical issues of software protection, even though the distribution form has been changed from ROM to magnetic media.

l.c.

" We believe there is direct violation of our international copyrights for the BASIC Programmer's ToolKit^tm,"

l.c.

(more)

Nestar Systems Inc.
NE 020
Feb. 14, 1980

{The sale of unauthoryed, modified copies of our product concerns us: it is bad for Nestar in particular, and for the microprocessor industry in general} good

commented Dr. Harry J. Saal, Nestar's president. " We ~~would not be announcing~~ think the ToolKit is such a phenomenal success ~~of our~~ ~~ToolKit if it weren'~~ because it'st a good value, and ~~gave~~ gives computer programmers what they want in reliable, well-designed software tools. We are now very concerned for our reputation, and for the micocomputer industry as well, by the sale of unauthorized, modified copies of our product."

I'd suggest wording above; it's better. 7

l.c.

The unauthorized copy ~~contains~~ IS Nestar's code which ~~someone has attempted to~~ has been incorrectly relocated to a ~~new~~ RAM memory location: under certain operating conditions, this may cause the running program to malfunction, and alter~~ing~~ or los~~ing~~e the user's data. In addition, one of the Nestar ToolKit commands no longer functions correctly. ~~A final~~ The only other change is the replacement of the Nestar Systems Incorporated copyright notice with the name of the company offering the unauthorized copies.

l.c.

"Nestar Systems, " Dr. Saal stressed, "is taking immediate action to prevent this unfair and illegal practice~~s~~ from continuing. We will pursue all remedies available to us under our international copyrights and trade agreements to stop the sale of any ~~and all~~ unauthorized ToolKit tm copies. Moreover, we encourage other manufacturers and dealers to join us in aggressively pursuing firms engaged in similar acts."

l.c.

(more)

The recent, spectacular growth in microcomputer sales makes issues of firmware, software, and hardware quality very critical. " Right now, product integrity and dependability are essential: suppliers' reputations and their sales success will depend on products performing as promised. Dealers and users are unknowingly taking risks with their programs when they use an unauthorized copy. "

Introduced in September, 1979, the BASIC Programmer's

l.c. ToolKit tm is distributed throughout North America, England, Western Europe, Japan, South Africa, Malaysia, (for real??)

l.c. and Australia. The ~~Available in~~ ROM based ~~form, the~~ ToolKit ~~also~~ is ~~includes~~ accompanied by an extensive 38 page user's manual.

In addition, ~~And~~ the substantial investment necessary to produce quality software will not be made ~~unless~~ if ~~this~~ illegal pirating becomes widespread

<u>Appendix 6</u>

excerpt from a Nestar Systems
internal operations report

software and utilities. The NFS station is designed to support multiuser applications and various software environments. The Nestar hardware components supported are:

- a floppy disk subsystem, supporting two double sided single density eight inch disk drives
- a hard disk subsystem, supporting one or two Winchester sealed disks with either 16.5MB or 33MB available capacity
- a streaming 1/4 inch tape cartridge subsystem for archival backup of disk storage, (holding 20MB per cartridge, taking 12 minutes to copy disk to tape)
- a real time clock/calendar (with battery backup) providing a central date, time of day and timestamping facility for the system.

The file server hardware is supported by a set of software components for managing these resources in a shared environment. The file server system software is composed of approximately 70,000 statements of high level language code (Pascal), with a small fraction of machine language device drivers. The user station software integration provides a sharable, protected environment for applications. Multiple file servers may be connected to the same network, thus providing for users who require more than the 66MB maximum storage per file server.

In additional, Nestar offers several packages which specifically support the Model A environment. These include "The Messenger", an interoffice electronic mail and memo facility, and a shared printer server. A full complement of utility programs and subroutines are provided for users to handle the network facilities.

Cluster/One Model One

The Cluster/One Model One is an early Nestar product offering which predates the Model A. It provides a small shared disk environment for a collection of microcomputers from Commodore, Apple and Tandy. It supports two floppy disks which can hold programs written in Microsoft Basic. No other languages, or data file storage is supported. The system was designed for use in low end educational environments, such as elementary or high schools.

The system is still available but no longer marketed actively by Nestar. A small number of residual orders and upgrades come in essentially unsolicited.

Consumer products

The Basic Programmer's Toolkit is a ROM hardware based upgrade to the Commodore Basic language which is part of the PET microcomputer. It is a low cost consumer item, and Nestar has sold over 25,000 units around the world. Nestar (under the name "Palo Alto ICs, a Division of Nestar Systems") will continue to market this product.

Nestar is currently investigating a follow-on product which provides similar function for the Commodore Basic contained in the VIC-20, a new under $300

Appendix 7

Nestar Systems inter-office memo

25 July 1980

from

Len Shustek (LJS)

to

Harry Saal (HJS)

Nick Fortis (NAF)

Don Anderson (DCA)

To: HJS, NAF, DCA
From: LJS
Re: Toolkit ads and the Nestar image
Date: 25 July 80

I am compelled to document what is evidently a minority opinion with regard to Toolkit advertising and the Nestar image.

I have in the past and still do now feel strongly that we need to do whatever we can to avoid the add-on peripheral gadget maker image. I think it is in conflict with the image we would like to project as the producer of networked systems of computers.

It is especially bad to be in the position of marketing a PET peripheral because, like it or not, the PET has the reputation of being a toy or kiddie computer. That public perception stems from the original 8K version with the laughable keyboard, and has not been altered by the force of Commodore´s reputation and less than professional marketing. I am not impugning the quality of our Toolkit, but we subject to guilt by association.

I do not object to a stronger association of Nestar with Palo Alto ICs, insofar as Nestar can capitalize on (or at least not suffer from) money spent for Toolkit advertising. I think it would be a big mistake, however, to run ads under the Nestar name with no indication that the Toolkit is not our main business.

I have no doubt that the ads would be professional and dignified, but the conclusion reached by someone who doesn´t know us already (and that is almost everyone) will be that Nestar is the company which makes Toolkits. If and when they find out that we also make bigger stuff, the impression may well be of a widget maker which is overreaching its grasp trying to get into the professional´s market.

A (WEAK) EXAMPLE

Paratronics is a company which started about 5 years ago by building very low cost logic analyzers for the hobby market. They established a modest reputation in that area, but were never taken seriously by the "professional" test instrument market (the HP/BIOMATION crowd).

They have since upgraded the equipment they make, to the point that their $9ØØØ analyzer is better than any other on the market. They still suffer, however, from their indentification as a seller to hobbyists; competing reps use prior products to knock the company, and even engineers (Bob Shopmeyer, for example) dismiss new products as inadequate unless forcibly shown otherwise. What has happened is that the default mindset is negative rather than neutral.

WHAT TO DO

As I said, I don't see much wrong with the association between Nestar and PAIC, and I think the proper way to advertise the Toolkit is as we have done: "Palo Alto ICs, A Division of Nestar Systems, Incorporated". We can argue about type sizes; we have certainly kept the "Nestar" inconspicuous in the past and could decide to change that. There is a qualitative difference in effect, however, from marketing it as a (the?) product of Nestar Systems.

From an operational standpoint it is also important to have a separate phone number for the retail trade (Toolkit orders and inquiries) so that the main switchboard is not clogged with them. It should be staffed during regular hours by a Toolkit person or persons. I don't want to answer Toolkit calls all night long.

THE MORAL

As long as we don't have a public image yet established, we have the luxury of building it as we would like it to be. We have the long lever arm now. If we start in the wrong direction, the damage will be difficult or impossible to undo.